

# INTERNATIONAL STANDARD

**Industrial communication networks – Fieldbus specifications –  
Part 6-2: Application layer protocol specification – Type 2 elements**

IECNORM.COM : Click to view the full PDF of IEC 61158-6-2:2023



**THIS PUBLICATION IS COPYRIGHT PROTECTED**  
**Copyright © 2023 IEC, Geneva, Switzerland**

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

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

Tel.: +41 22 919 02 11  
[info@iec.ch](mailto:info@iec.ch)  
[www.iec.ch](http://www.iec.ch)

**About the IEC**

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

**About IEC publications**

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

**IEC publications search - [webstore.iec.ch/advsearchform](http://webstore.iec.ch/advsearchform)**

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 It also gives information on projects, replaced and withdrawn publications.

**IEC Just Published - [webstore.iec.ch/justpublished](http://webstore.iec.ch/justpublished)**

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

**IEC Customer Service Centre - [webstore.iec.ch/csc](http://webstore.iec.ch/csc)**

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

**IEC Products & Services Portal - [products.iec.ch](http://products.iec.ch)**

Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

**Electropedia - [www.electropedia.org](http://www.electropedia.org)**

The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IECNORM.COM : Click to view the full PDF IEC 61158-6-2:2023

# INTERNATIONAL STANDARD

---

**Industrial communication networks – Fieldbus specifications –  
Part 6-2: Application layer protocol specification – Type 2 elements**

INTERNATIONAL  
ELECTROTECHNICAL  
COMMISSION

---

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-6631-1

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

## 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 .....                                                                                               | 251 |

|                                                                                                                                       |     |
|---------------------------------------------------------------------------------------------------------------------------------------|-----|
| 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<br>(Parameter object stub) ..... | 110 |
| Table 142 – Instance level object/service specific response data of Get_Attributes_All<br>(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](http://www.iec.ch/members_experts/refdocs). The main document types developed by IEC are described in greater detail at [www.iec.ch/publications](http://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](https://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.

IECNORM.COM : Click to view the full PDF of IEC 61158-6-2:2023

## 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.

IECNORM.COM : Click to view the full PDF of IEC 61158-6-2:2023

## **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

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

##### 3.1.2 Terms and definitions from ISO/IEC 9545

- application-association
- application-context
- application context name
- application-entity-invocation
- application-entity-type
- application-process-invocation
- application-process-type
- application-service-element
- 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 <sup>1</sup> – 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

---

<sup>1</sup> 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 that 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](http://www.odva.org)>).

### 3.4 Abbreviated terms and symbols

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <b>ASCII</b>      | American Standard Code for Information Interchange                    |
| <b>CID</b>        | connection ID                                                         |
| <b>CAN</b>        | Controller Area Network (see ISO 11898-1)                             |
| <b>CM_API</b>     | actual packet interval                                                |
| <b>CM_RPI</b>     | requested packet interval                                             |
| <b>DLL</b>        | data-link layer                                                       |
| <b>DSCP</b>       | DiffServ Codepoint                                                    |
| <b>IGMP</b>       | Internet Group Management Protocol (see IETF RFC 1112, IETF RFC 2236) |
| <b>IP</b>         | Internet Protocol (see IETF RFC 791)                                  |
| <b>IPv4</b>       | Internet Protocol version 4 (see IETF RFC 791)                        |
| <b>IPv6</b>       | Internet Protocol version 6 (see IETF RFC 791)                        |
| <b>O2T or O⇒T</b> | originator to target (connection parameters)                          |
| <b>OSI</b>        | open systems interconnection (see ISO/IEC 7498-1)                     |
| <b>PDU</b>        | protocol data unit                                                    |
| <b>PHB</b>        | per-hop behavior                                                      |
| <b>PTP</b>        | Precision Time Protocol (see IEC 61588)                               |
| <b>QoS</b>        | quality of service                                                    |
| <b>Rcv</b>        | receive                                                               |
| <b>Rx</b>         | receive                                                               |
| <b>SDU</b>        | service data unit                                                     |
| <b>SEM</b>        | state event matrix                                                    |
| <b>STD</b>        | state transition diagram, used to describe object behavior            |
| <b>T2O or T⇒O</b> | target to originator (connection parameters)                          |
| <b>TCP</b>        | Terminal Control Protocol (see IETF RFC 793)                          |
| <b>ToS</b>        | type of service                                                       |
| <b>TPDU</b>       | transport protocol data unit                                          |
| <b>TUI</b>        | table unique identifier                                               |
| <b>Tx</b>         | transmit                                                              |
| <b>UDP</b>        | User Datagram Protocol (see IETF RFC 768)                             |
| <b>Xmit</b>       | 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"> <li>– 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;</li> <li>– 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.</li> </ul> |
| 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<br>(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.

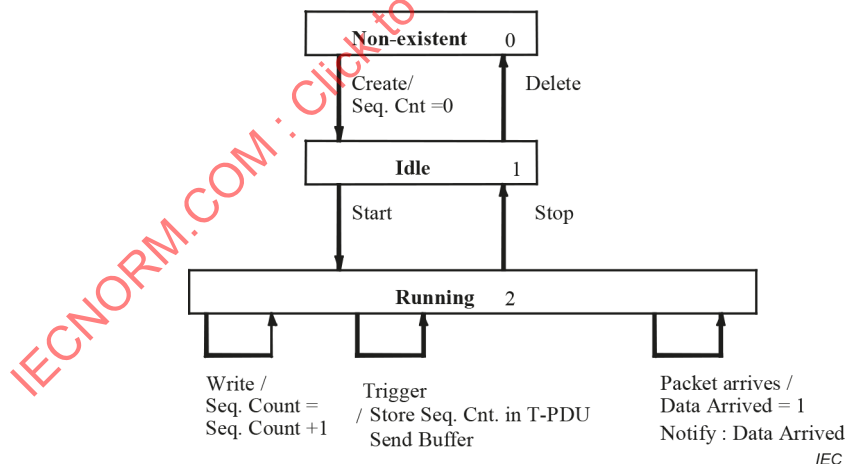


Figure 3 – Example of an STD

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 <sup>a</sup>                                                                                                             |      |                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|------|--------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                          | State 1                                                                                                                        | .... | State n                                                                                                                        |
| Event A description                                                                                                      | Function triggered by event A in State 1 (if any)<br>Notification to FAL user (if any)<br>Transition to another state (if any) |      | Function triggered by event A in State n (if any)<br>Notification to FAL user (if any)<br>Transition to another state (if any) |
| .....                                                                                                                    | .....                                                                                                                          |      | .....                                                                                                                          |
| Event X description                                                                                                      | Function triggered by event X in State 1 (if any)<br>Notification to FAL user (if any)<br>Transition to another state (if any) |      | Function triggered by event X in State n (if any)<br>Notification to FAL user (if any)<br>Transition to another state (if any) |
| <sup>a</sup> 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.<br>Return response       | Validate/service the request.<br>Return response       |
| Set_Attribute_Single | Error: Object does not exist. (General Error Code 0x16) | Validate/service the request.<br>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 TPDUs shall use the `DLL_fixed_request.req` and `DLL_fixed_request.ind` service primitives of the data-link layer, respectively. The packet parameter of these services shall be prefixed with the following header to form the Transport PDU before sending to the data-link layer. The format of the UCMM\_PDU Header is shown in Table 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  $(8 \mid \text{timeout} \& 7) \ll (\text{timeout} \gg 3 \& 31)$ .

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

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

Only one message shall be outstanding for a given RECORD.

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

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

#### 4.1.4 Transport\_Headers

##### 4.1.4.1 General

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

##### 4.1.4.2 Class 0 (null or base)

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

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

#### 4.1.5.3.3 Format of Forward\_Open request

The format of the Forward\_Open request TPDU shall be of the form shown in Table 15.

**Table 15 – Forward\_Open request format**

| Parameter name                                                                             | Format       |
|--------------------------------------------------------------------------------------------|--------------|
| priority_and_tick                                                                          | SWORD        |
| connection time-out ticks                                                                  | USINT        |
| O2T_CID                                                                                    | UDINT        |
| T2O_CID                                                                                    | UDINT        |
| connection_serial_number                                                                   | UINT         |
| originator_vendor_ID                                                                       | UINT         |
| originator_serial_number                                                                   | UDINT        |
| connection timeout multiplier                                                              | USINT        |
| reserved[3]                                                                                | USINT        |
| O2T_CM_RPI                                                                                 | UDINT        |
| O2T_connection_parameters <sup>a</sup>                                                     | UINT         |
| T2O_CM_RPI                                                                                 | UDINT        |
| T2O_connection_parameters <sup>a</sup>                                                     | UINT         |
| xport_type_and_trigger                                                                     | SWORD        |
| connection_path_size                                                                       | USINT        |
| connection_path                                                                            | Padded EPATH |
| <sup>a</sup> Connection type in this parameter shall be 00 (NULL) for a null Forward_Open. |              |

#### 4.1.5.3.4 Format of Forward\_Open response if success

If the status parameter of the CM\_open\_response is zero (no error), the format of the Forward\_Open response TPDU shall be of the form shown in Table 16.

**Table 16 – Forward\_Open\_Good response format**

| Parameter name                            | Format |
|-------------------------------------------|--------|
| O2T_CID                                   | UDINT  |
| T2O_CID                                   | UDINT  |
| connection_serial_number                  | UINT   |
| originator_vendor_ID                      | UINT   |
| originator_serial_number                  | UDINT  |
| O2T_CM_API                                | UDINT  |
| T2O_CM_API                                | UDINT  |
| Application_reply_size; (in 16-bit words) | USINT  |
| Reserved                                  | USINT  |
| application_reply[]                       | UINT   |

If a Connection object is instantiated to support this connection:

- the values of O2T\_CM\_API and T2O\_CM\_API are used for the Connection object Expected\_packet\_rate attribute, after converting to milliseconds.
- the Expected\_packet\_rate will not be used for the Connection object Inactivity/Watchdog Timer. See 4.1.6.2 for connection timeout handling.

#### 4.1.5.3.5 Format of Forward\_Open response if failure

If the `status` parameter of the `CM_open_response` is not zero (error), the format of the Forward\_Open response TPDU shall be of the form shown in Table 17.

**Table 17 – Forward\_Open\_Bad response format**

| Parameter name                               | Format    |
|----------------------------------------------|-----------|
| connection_serial_number                     | UINT      |
| originator_vendor_ID                         | UINT      |
| originator_serial_number                     | UDINT     |
| remaining_path_size                          | STRUCT of |
| number of words in connection_path parameter | USINT     |
| reserved (shall be set to 0)                 | SWORD     |

This format shall be used for all failures of the connection system. The requested connection shall not be established, and the object specific status words shall contain information about the reason for the failure.

The `remaining_path_size` field is required if the error was detected by a router, optional if detected by the target. It shall contain the length of the path at the point the connection failed, i.e. the number of words in the `connection_path` parameter of the request as received by the device that detects the error.

In the failure response, the `remaining_path_size` shall be the “pre-stripped” size. This shall be the size of the path when the node first receives the request and has not yet started processing it. A target node may return either the “pre-stripped” size or 0 for the `remaining_path_size`.

A duplicate Forward\_Open service shall be defined as a Forward\_Open service whose `originator_vendor_ID`, `connection_serial_number`, and `originator_serial_number` match an existing connection's parameters. If the duplicate Forward\_Open service is a null Forward\_Open service (defined as the connection type in both the O2T and T2O network connection parameter fields are NULL), then the Forward\_Open service shall be forwarded to the application for further processing. Null Forward\_Open requests may be used to reconfigure the connection. The Connection Manager in the routers need not allocate additional resources for a duplicate Forward\_Open request since the resources have already been allocated. If the duplicate Forward\_Open request is not NULL, then an general status = 0x01, extended status = 0x0100 shall be returned.

#### 4.1.5.4 Large\_Forward\_Open

##### 4.1.5.4.1 General

The Large\_Forward\_Open shall have the same function and same behavior as the Forward\_Open except that it shall allow the establishment of connections larger than 511 octets.

The response to the Large\_Forward\_Open\_Request\_PDU is an Large\_Forward\_Open\_ResponsePDU. All other requirements are identical to those specified for the Forward\_Open service in 4.1.5.3.

##### 4.1.5.4.2 Format of Large\_Forward\_Open request

The format of the Large\_Forward\_Open request TPDU shall be of the form shown in Table 18.

**Table 18 – Large\_Forward\_Open request format**

| Parameter name                | Format       |
|-------------------------------|--------------|
| priority_and_tick             | SWORD        |
| connection time-out ticks     | USINT        |
| O2T_CID                       | UDINT        |
| T2O_CID                       | UDINT        |
| connection_serial_number      | UINT         |
| originator_vendor_ID          | UINT         |
| originator_serial_number      | UDINT        |
| connection timeout multiplier | USINT        |
| reserved[3]                   | USINT        |
| O2T_CM_RPI                    | UDINT        |
| O2T_ex_connection_size        | UINT         |
| O2T_connection_parameters     | UDINT        |
| T2O_CM_RPI                    | UDINT        |
| T2O_ex_connection_size        | UINT         |
| T2O_connection_parameters     | UDINT        |
| xport_type_and_trigger        | SWORD        |
| connection_path_size          | USINT        |
| connection_path               | Padded EPATH |

The connection size field in the two connection\_parameters shall be reserved in this case and set to zero. The ex\_connection\_size parameters shall be used instead to specify the maximum size of the connection (up to 65 535).

#### 4.1.5.4.3 Format of Large\_Forward\_Open response if success

If the `status` parameter of the `CM_open_response` is zero (no error), the format of the `Large_Forward_Open` response TPDU shall be of the form shown in Table 19.

**Table 19 – Large\_Forward\_Open\_Good response format**

| Parameter name                            | Format |
|-------------------------------------------|--------|
| O2T_CID                                   | UDINT  |
| T2O_CID                                   | UDINT  |
| connection_serial_number                  | UINT   |
| originator_vendor_ID                      | UINT   |
| originator_serial_number                  | UDINT  |
| O2T_CM_API                                | UDINT  |
| T2O_CM_API                                | UDINT  |
| Application_reply_size; (in 16-bit words) | USINT  |
| Reserved                                  | USINT  |
| application_reply[]                       | UINT   |

#### 4.1.5.4.4 Format of Large\_Forward\_Open response if failure

If the `status` parameter of the `CM_open_response` is not zero (error), the format of the `Large_Forward_Open` response TPDU shall be of the form shown in Table 20.

**Table 20 – Large\_Forward\_Open\_Bad response format**

| Parameter name                               | Format    |
|----------------------------------------------|-----------|
| connection_serial_number                     | UINT      |
| originator_vendor_ID                         | UINT      |
| originator_serial_number                     | UDINT     |
| remaining_path_size                          | STRUCT of |
| number of words in connection_path parameter | USINT     |
| reserved (shall be set to 0)                 | SWORD     |

Either a target device or an intermediate router shall return an Invalid Connection Size extended error code, along with the maximum connection size supported extended status word, if the connection size requested is not supported while processing a `Large_Forward_Open` request.

#### 4.1.5.5 Forward\_Close

##### 4.1.5.5.1 General

The `Forward_Close` request shall remove a connection from all the nodes participating in the original connection. The `Forward_Close` shall be sent between Connection Managers as specified in the `connection_path`. The `Forward_Close` request shall cause all non-shared resources in all nodes participating in the connection to be deallocated, including connection IDs, link transmit time, and internal memory buffers. Shared resources are those still in use by other connection(s) due to multicast.

If a router cannot find the connection that is to be closed (it could have timed out at the node), the Forward\_Close request shall still be forwarded to downstream nodes or the target application (via the CM\_close\_indication).

NOTE The Forward\_Close is always forwarded to allow downstream nodes or the target application to release any resources that were allocated for the connection.

#### 4.1.5.5.2 Format of Forward\_Close request

The format of the Forward\_Close request shall be of the form shown in Table 21.

The path specified in the MR\_Request\_Header for the Forward\_Close 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\_Close service.

**Table 21 – Forward\_Close request format**

| Parameter name                | Format          |
|-------------------------------|-----------------|
| connection_priority/tick time | SWORD           |
| connection_timeout            | USINT           |
| connection_serial_number      | UINT            |
| originator_vendor_ID          | UINT            |
| originator_serial_number      | UDINT           |
| connection_path_size          | USINT           |
| Reserved                      | USINT           |
| connection_path               | Padded<br>EPATH |

#### 4.1.5.5.3 Format of Forward\_Close response if success

The Forward\_Close response shall be sent between Connection Managers in response to a Forward\_Close request, and shall contain the status of the close request. If the status parameter of the CM\_close\_response is zero (no error), the format of the Forward\_Close response TPDU shall be of the form shown in Table 22.

**Table 22 – Forward\_Close\_Good response format**

| Parameter name                          | Format |
|-----------------------------------------|--------|
| connection serial number                | UINT   |
| originator_vendor ID                    | UINT   |
| originator serial number                | UDINT  |
| application_reply_size; in 16-bit words | USINT  |
| Reserved                                | USINT  |
| application_reply[]                     | UINT   |

A successful Forward\_Close response shall be returned when the close has been acknowledged by all the nodes from this point in the path to the end of the path. The response shall indicate the connection\_serial\_number, originator\_vendor\_ID, and originator\_serial\_number of the closed connection. Once the response indicating success has been received, the connection shall be closed from this point in the path to the end. The target application may optionally pass back application specific information in the successful close response. If no application\_reply information is contained in the service the application\_reply\_size shall be zero.

#### 4.1.5.5.4 Format of Forward\_Close response if failure

If the `status` parameter of the `CM_close_response` is not zero (error), the format of the `Forward_Close` response shall be of the form shown in Table 23.

**Table 23 – Forward\_Close\_Bad response format**

| Parameter name                                               | Format    |
|--------------------------------------------------------------|-----------|
| <code>connection_serial_number</code>                        | UINT      |
| <code>originator_vendor_ID</code>                            | UINT      |
| <code>originator_serial_number</code>                        | UDINT     |
| <code>remaining_path_size</code>                             | STRUCT of |
| number of words in<br><code>connection_path</code> parameter | USINT     |
| reserved (shall be set to 0)                                 | SWORD     |

The object specific status words shall contain information about the reason for the failure. The `remaining_path_size` field is required if the error was detected by a router, optional if detected by the target. It shall contain the path size from the point at which the close connection failed, i.e. the number of words in the `connection_path` parameter of the request as received by the device that detects the error.

In the failure response, the `remaining_path_size` shall be the “pre-stripped” size. This shall be the size of the path when the node first receives the request and has not yet started processing it. A target node shall return either the “pre-stripped” size or 0 for the `remaining_path_size`.

#### 4.1.5.6 Unconnected\_Send

##### 4.1.5.6.1 General

The `Unconnected_Send` service shall allow an application to send a message to a device without first setting up a connection. The `Unconnected_Send` service shall use the Connection Manager object in each router to forward the message and to remember the return path. The UCMM of each link shall be used to forward the request from Connection Manager to Connection Manager just as it is for the `Forward_Open` service; however. No connection shall be built. The `Unconnected_Send` service shall be sent to the local Connection Manager and shall be sent between routers. When a router removes the last port segment, the embedded message request shall be formatted as an `OM_Request` and sent to the port and link address of the last port segment using the UCMM for that link type.

NOTE The target node never sees the `Unconnected_Send` service but only the embedded message request arriving via the UCMM.

##### 4.1.5.6.2 Format of Unconnected\_Send request

The format of the `Unconnected_Send` request shall be of the form shown in Table 24.

**Table 24 – Unconnected\_Send request format**

| Parameter name      | Format       | Description                                                                                                        |
|---------------------|--------------|--------------------------------------------------------------------------------------------------------------------|
| priority/tick time  | SWORD        | Used to calculate request timeout information                                                                      |
| conn time-out ticks | USINT        |                                                                                                                    |
| Message_size        | UINT         | Number of octets in the embedded Message Router PDU                                                                |
| Message Router PDU  | OM_Request   | Embedded Messenger Router PDU                                                                                      |
| pad                 | USINT        | Only present if Message_size is an odd value. This allows the remaining fields to be aligned on a 16-bit boundary. |
| Route_path_size     | USINT        | Number of 16-bit words in the route_path field                                                                     |
| Reserved            | USINT        | Reserved, shall be set to zero (0)                                                                                 |
| route_path          | Padded EPATH | Indicates the route to the remote target device                                                                    |

The path specified in the MR\_Request\_Header for the Unconnected\_Send service shall not contain any Electronic Key segment. Electronic key segments may be included in the path specified in the MR\_Request\_Header of the embedded message request and in the route\_path parameter of the Unconnected\_Send service.

The route path shall be of the form:

```

[Electronic Key segment1]      } Port Segment Group for 1st router
Port segment1                  }

[[Electronic Key segment2]     } Port Segment Group for 2nd router
Port segment2                  }

...

[[Electronic Key segmentN]     } Port Segment Group for Nth router
Port segmentN                  }

```

All segments listed in brackets [...] are optional. If the Electronic Key segment is present the target shall evaluate the key. See 4.1.9.3 for the definition of Port segment and 4.1.9.4.2 for the definition of Electronic Key segment.

#### 4.1.5.6.3 Unconnected\_Send response

The Unconnected\_Send response shall be generated by the last router from the UCMM response generated by the target node or by a router as the result of a UCMM time-out, a problem with the embedded message, or a problem with the Unconnected Service Request itself. The packet shall be routed from router to router using the information stored when the Unconnected\_Send request was processed. The response shall contain a header with status information about the request and a variable length response generated by the target node.

#### 4.1.5.6.4 Format of Unconnected\_Send response if success

This format shall be used when a successful Unconnected\_Send response is received. The structure of the response is shown in Table 25.

**Table 25 – Unconnected\_Send\_Good response format**

| Parameter name           | Format |
|--------------------------|--------|
| application_response [ ] | OCTET  |

The `application_response` field contains the data returned by the target device/object in the `Service_ResponsePDU` for the embedded request. If the `Service_ResponsePDU` returned by the target device/object did not contain any data, then this field shall be empty.

EXAMPLE This field would contain Attribute Data in response to an embedded `Get_Attribute_Single` request.

#### 4.1.5.6.5 Format of Unconnected\_Send response if failure

In case of a failure of the `Unconnected_Send` service, the Service code returned in the `MR_Response_Header` for the `Unconnected_Send` service may correspond either to the Service code sent inside the `Unconnected_Send` service data (embedded message request) or to the `Unconnected_Send` Service code itself:

- when a router detects the error, the `Unconnected_Send` response Service code (0xD2) shall be returned in the `MR_Response_Header`;
- when the target device detects the error, the response Service code for the embedded message request shall be returned in the `MR_Response_Header`.

EXAMPLE If the target device detects the error and the requested service was a `Get_Attribute_Single`, then the Service code in the `MR_Response_Header` will be 0x8E.

This is advantageous to originators and routers because:

- of the two service codes (`Unconnected_Send` and embedded), the service code that failed is returned to the originator;
- routers are not required to parse the Service code of the embedded message within the `Unconnected_Send` request.

The format of the `Unconnected_Send` response service for a failure shall be of the form shown in Table 26.

**Table 26 – Unconnected\_Send\_Bad response format**

| Parameter name                                       | Format    |
|------------------------------------------------------|-----------|
| <code>remaining_path_size</code>                     | STRUCT of |
| number of words in <code>route_path</code> parameter | USINT     |
| reserved (shall be set to 0)                         | SWORD     |

The `remaining_path_size` field shall only be present if the error was detected by a router. It indicates the number of words in the `route_path` parameter of the request as received by the router that detects the error.

#### 4.1.5.7 Get\_Connection\_Data

##### 4.1.5.7.1 General

This service shall return the parameters associated with a specified connection number. The connection number may be different from device to device even for the same connection. The connection number corresponds to the offset into the Connection Manager attribute that enumerates the status of the connections.

NOTE This service can be used for network diagnostics.

##### 4.1.5.7.2 Format of Get\_Connection\_Data request

The format of the `Get_Connection_Data` request shall be of the form shown in Table 27.

**Table 27 – Get\_Connection\_Data request format**

| Parameter name    | Format |
|-------------------|--------|
| connection_number | UINT   |

#### 4.1.5.7.3 Format of Get\_Connection\_Data response

The format of the Get\_Connection\_Data response TPDU shall be of the form shown in Table 28.

**Table 28 – Get\_Connection\_Data response format**

| Parameter name                    | Format |
|-----------------------------------|--------|
| connection_number                 | UINT   |
| connection_state                  | UINT   |
| originator_port                   | UINT   |
| target_port                       | UINT   |
| connection_serial_number          | UINT   |
| originator_vendor_ID              | UINT   |
| originator_serial_number          | UDINT  |
| originator_O2T_CID                | UDINT  |
| target_O2T_CID                    | UDINT  |
| O2T_connection timeout multiplier | USINT  |
| reserved1[3]                      | USINT  |
| originator_O2T_CM_RPI             | UDINT  |
| originator_O2T_CM_API             | UDINT  |
| originator_T2O_CID                | UDINT  |
| target_T2O_CID                    | UDINT  |
| T2O_connection timeout multiplier | USINT  |
| reserved2[3]                      | USINT  |
| originator_T2O_CM_RPI             | UDINT  |
| originator_T2O_CM_API             | UDINT  |

#### 4.1.5.8 Search\_Connection\_Data

##### 4.1.5.8.1 General

This service shall return the parameters associated with a specified connection within a device identified by vendor and serial number.

NOTE This service can be used for network diagnostics.

##### 4.1.5.8.2 Format of Search\_Connection\_Data request

The format of the Search\_Connection\_Data request shall be of the form shown in Table 29.

**Table 29 – Search\_Connection\_Data request format**

| Parameter name           | Format |
|--------------------------|--------|
| connection_serial_number | UINT   |
| originator_vendor_ID     | UINT   |
| originator_serial_number | UDINT  |

**4.1.5.8.3 Format of Search\_Connection\_Data response**

The format of the Search\_Connection\_Data response shall be the same as the response from the Get\_Connection\_Data service (see 4.1.5.7).

**4.1.5.9 Get\_Connection\_Owner****4.1.5.9.1 General**

The Get\_Connection\_Owner service of the Connection Manager shall return data about the connection(s) that own(s) a particular object. It shall be implemented in any device that accepts redundant connections.

**4.1.5.9.2 Format of Get\_Connection\_Owner request**

The format of the Get\_Connection\_Owner request TPDU shall be of the form shown in Table 30:

**Table 30 – Get\_Connection\_Owner request format**

| Parameter name | Format       |
|----------------|--------------|
| reserved       | USINT        |
| path_size      | USINT        |
| path[]         | Padded EPATH |

The reserved field shall be set to zero.

**4.1.5.9.3 Format of Get\_Connection\_Owner response**

The format of the Get\_Connection\_Owner response TPDU shall be of the form shown in Table 31.

**Table 31 – Get\_Connection\_Owner response format**

| Parameter name             | Format | Description                                                                                                                                                                                                                               |
|----------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| number_of_connections      | USINT  |                                                                                                                                                                                                                                           |
| number_claiming_ownership  | USINT  |                                                                                                                                                                                                                                           |
| number_ready_for_ownership | USINT  |                                                                                                                                                                                                                                           |
| last_action                | USINT  | Ownership status:<br>0 = there is no owning connection.*<br>1 = the owning connection is in idle mode<br>2 = the owning connection is in run mode<br>3 to 254 = reserved<br>255 = the implementation does not support one of these fields |
| connection_serial_number   | UINT   | If there is no owner the value shall be 0                                                                                                                                                                                                 |
| originator_vendor_ID       | UINT   | If there is no owner the value shall be 0                                                                                                                                                                                                 |
| originator_serial_number   | UDINT  | If there is no owner the value shall be 0                                                                                                                                                                                                 |

## 4.1.6 CM PDU components

### 4.1.6.1 Network connection parameters

#### 4.1.6.1.1 Format

Network connection parameters shall be provided as a single 16-bit word that contains five fields, shown in Figure 4. The fields within the 16-bit word shall indicate

- the type of network connection desired;
- whether the buffer size is variable or fixed;
- the priority of the connection;
- the size of the connection buffer required.

The size of the connection buffer shall include any transport header.

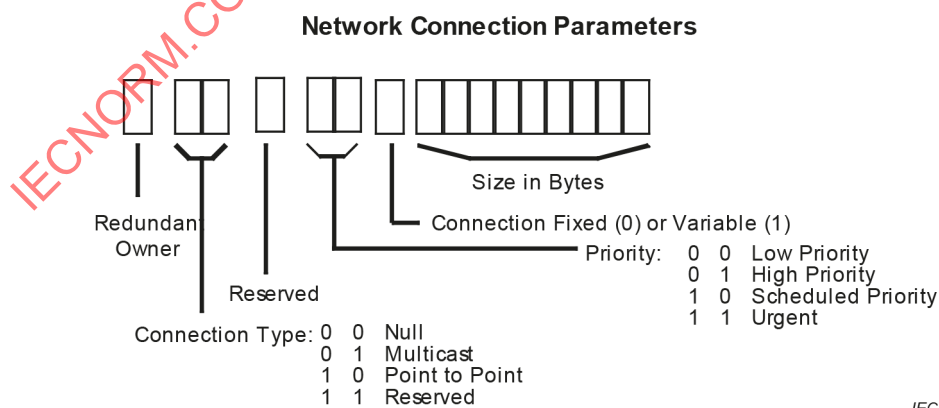


**Figure 4 – Network connection parameters**

#### 4.1.6.1.2 Connection type

The connection type shall be one of NULL, MULTIPOINT, or POINT2POINT. The NULL type shall indicate that no network connection is required, while MULTIPOINT and POINT2POINT

refer to the network connection types as defined in IEC 61158-5-2. The NULL connection type may be used to reconfigure a connection.

#### **4.1.6.1.3 Priority**

Priority shall be one of LOW, HIGH, SCHEDULED and URGENT, with URGENT being the highest priority (see 4.1.6.3 for detailed specification and usage).

#### **4.1.6.1.4 Connection fixed/variable**

The connection can be set up to use fixed or variable sized connections. With a fixed size connection, each transmission on the connection shall use the same size buffer. Otherwise, either a buffer overrun or underrun condition shall occur. With a variable sized connection, each transmission on the connection may send a variable amount of data up to the maximum size, which shall be specified when the connection is opened. Buffer underruns shall not be reported, but a buffer overrun can still occur if too much data is sent.

#### **4.1.6.1.5 Connection size**

The network connection size shall be the size of the buffer required for the connection, which includes the data and any transport header. For a variable sized connection, the size shall be the maximum size of the buffer for any transfer. The actual size of the transfer for a variable connection shall be equal to or less than the size specified for the network connection. The maximum buffer size shall be dependent on the links that the connection traverses.

#### **4.1.6.1.6 Redundant owner**

The redundant owner bit in the O⇒T direction shall be set (= 1) to indicate that more than one owner is permitted to make a connection simultaneously. The bit shall be clear (= 0) to indicate an exclusive-owner, input only or listen-only connection.

#### **4.1.6.1.7 Reserved parameters**

Reserved bits shall be clear (= 0).

#### **4.1.6.2 Requested and actual packet interval (CM\_RPI and CM\_API)**

The connection originator provides a requested time between productions for each direction of the connection. The target provides the actual time between productions. When a connection traverses some Type 2 networks, the actual time may be further adjusted as specified in the corresponding Type 2 network communication profile.

The requested packet interval (CM\_RPI) shall be the requested time between packets in microseconds. The format of the CM\_RPI shall be a 32-bit integer in microseconds.

The actual packet interval (CM\_API) shall be the maximum time between packets in microseconds. The CM\_API is equivalent to the Transmission Trigger Timer value specified in IEC 61158-5-2, 6.2.3.2.1.7. The format of the CM\_API shall be a 32-bit integer in microseconds.

The influence of the CM\_API and CM\_RPI values on the behavior of the connection is dependent on the Transport Class and Trigger definitions in 4.1.8.9.1.2. The CM\_RPI value shall be used to allocate capacity usage at each of the producing nodes. The allocation of capacity usage could have to be adjusted when the CM\_API is returned, since it is possible for the CM\_RPI and CM\_API to differ.

The CM\_API shall equal the CM\_RPI within the device's timer resolution limits, except if specified otherwise in the Type 2 network communication profile. If the CM\_API is not equal to the CM\_RPI it shall be faster than the CM\_RPI unless the resulting CM\_API would be 0, in which case the CM\_RPI cannot be supported and an error shall be returned, either:

- CM\_RPI Value(s) Not Acceptable, General Status 0x01, Extended Status 0x0112 (recommended);
- CM\_RPI Not Supported, General Status 0x01, Extended Status 0x0111.

The Inactivity/Watchdog time-out value at each of the intermediate and target nodes shall be set to the connection time-out multiplier times the CM\_RPI. If the resulting time-out is not achievable within the device's timer resolution limits, the time-out shall be rounded up the next value within the device's timer resolution limits.

#### 4.1.6.3 Function of priorities

The order of priorities, from highest to lowest are: urgent, scheduled, high and low.

While there are no specific implementation requirements for end devices to internally differentiate traffic with different priorities, devices are recommended to give preferential processing to higher priority connections over lower priority connections. At a minimum, devices are recommended to give higher priority to processing implicit messages over explicit messages.

The mapping of Type 2 Priorities to link priorities is specific to each Type 2 network communication profile.

##### ***Urgent priority***

Urgent priority shall only be used where defined explicitly in the Type 2 companion standards.

EXAMPLE Urgent priority is used for I/O connections established with the Motion Device Axis object specified in IEC 61800-7-202.

Nodes shall map the urgent priority to the corresponding network specific link priority.

##### ***Scheduled Priority***

Nodes shall map the scheduled priority to the corresponding network specific link priority.

##### ***High Priority***

Nodes shall map the high priority to the corresponding network specific link priority.

##### ***Low Priority***

Nodes shall map the low priority to the corresponding network specific link priority.

#### 4.1.6.4 Connection time-out multiplier

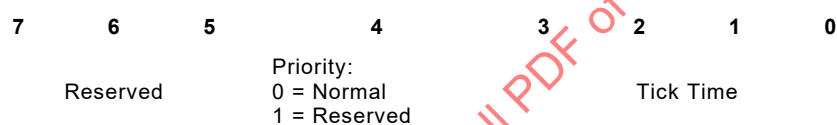
The Connection Time-out Multiplier specifies the multiplier applied to the CM\_RPI to obtain the connection time-out value. Device shall stop transmitting on a connection whenever the connection times out even if the pending close has been sent. The multiplier shall be as represented by Table 32.

**Table 32 – Time-out multiplier**

| Value    | Multiplier | Notes   |
|----------|------------|---------|
| 0        | × 4        | Default |
| 1        | × 8        |         |
| 2        | × 16       |         |
| 3        | × 32       |         |
| 4        | × 64       |         |
| 5        | × 128      |         |
| 6        | × 256      |         |
| 7        | × 512      |         |
| 8 to 255 | reserved   |         |

**4.1.6.5 Connection request priority and tick time**

The Priority/Tick\_time parameter determines the priority of the unconnected message and the time duration of a “tick” (Tick Time) specified in the Time-out\_ticks parameter. The bit fields are specified in Figure 5.

**Figure 5 – Priority/Tick\_time bit definition**

The Reserved and Priority fields shall be set to zero (0). The Tick Time shall be used in conjunction with the Time-out\_ticks parameter value to determine the total time out value.

The following formula is used:

$$\text{Actual Time Out value} = 2^{\text{Tick\_Time}} \times \text{Time\_out\_tick}$$

EXAMPLE If the tick time is 0000 (1 ms), a Time-out\_ticks value of 5 would translate into 5 ms. If the tick time is 0010 (4 ms), a Time-out\_ticks value of 5 would translate into 20 ms.

Table 33 shall determine the time between ticks.

**Table 33 – Tick time units**

| Tick time          |               |              |
|--------------------|---------------|--------------|
| Tick time (binary) | Time per tick | Max. time    |
| 0000               | 1 ms          | 255 ms       |
| 0001               | 2 ms          | 510 ms       |
| 0010               | 4 ms          | 1 020 ms     |
| 0011               | 8 ms          | 2 040 ms     |
| 0100               | 16 ms         | 4 080 ms     |
| 0101               | 32 ms         | 8 160 ms     |
| 0110               | 64 ms         | 16 320 ms    |
| 0111               | 128 ms        | 32 640 ms    |
| 1000               | 256 ms        | 65 280 ms    |
| 1001               | 512 ms        | 130 560 ms   |
| 1010               | 1 024 ms      | 261 120 ms   |
| 1011               | 2 048 ms      | 522 240 ms   |
| 1100               | 4 096 ms      | 1 044 480 ms |
| 1101               | 8 192 ms      | 2 088 960 ms |
| 1110               | 16 384 ms     | 4 177 920 ms |
| 1111               | 32 768 ms     | 8 355 840 ms |

**4.1.6.6 Connection request time-out ticks**

The Time-out ticks parameter shall be used to specify the amount of time the originating application shall wait for the transaction to be completed. When used with the Tick\_Time portion of the Priority/Tick\_time field, a total timeout value can be calculated.

**Timeout Algorithm**

The *Priority/Tick\_time* and *Time-out ticks* parameters are used to convey timeout information as an unconnected request flows through interconnecting devices. The originator states how long it will wait for a response (total round trip time).

Each router subtracts the approximate amount of time it needs to process and forward both the request and the associated response. If a router cannot determine specifically how much time to subtract, it shall subtract 512 ms.

When encoding the result of the subtraction, the router shall use the smallest *Tick\_time* possible (see 4.1.6.5) and ensure that the outgoing value is less than the received value. This provides the best granularity for decreasing the timeout as the message traverses the system. This also means that the *Tick\_time* can change as the message is passed through multiple hops.

Each router shall use the new value as a timer for the resources used to manage the unconnected message and also to check to see if a timeout is imminent, i.e. the remaining time is zero or less, before forwarding the message. If a timeout is imminent, the router returns a General Status code of 0x01 with Extended Status code of 0x204 (Unconnected Request Timed Out) error response immediately rather than just letting the originator timeout.

If a router can detect that the next hop device is not present (e.g. the router supports an active node table function), it returns a General Status code of 0x01 with Extended Status code of 0x0204 (Unconnected Request Timed Out) error response immediately.

When a router fails to receive a response in the remaining time, a General Status code of 0x01 with Extended Status code of 0x0204 (Unconnected Request Timed Out) error response is returned.

When a target receives a Forward\_Open, Large\_Forward\_Open or Forward\_Close request, the target may either:

- ignore the *Priority/Tick\_time* and *Time-out ticks* parameters and process the request, or;
- evaluate the *Priority/Tick\_time* and *Time-out ticks* parameters and return a General Status code of 0x01 with Extended Status code of 0x0204 (Unconnected Request Timed Out) error response immediately if it knows it will not have enough time to complete the request.

If a target evaluates the *Priority/Tick\_time* and *Time-out ticks* parameters:

- Adding the timing evaluation to a target as part of a product update needs to be carefully considered. An updated product shall use its best case time for comparison. Thus, an updated target should always accept connections that previously succeeded.
- Supporting the timing evaluation in a target as part of a new product shall use its worst case time for comparison. The result is that the target (if present) sends the timeout response instead of any router device.
- If a new or updated product can emulate one or more existing device(s), it shall perform the timing behavior of the device indicated in the compatible key of the request.

In all error cases, the node that generates the error response includes the Remaining Path Size parameter and every node that receives it releases all resources associated with that unconnected message. The Remaining Path Size in the error response facilitates the ability to know how far along the route the request progressed before encountering the error.

When an “Unconnected Request Timed Out” error is encountered, it is most likely due to one of the devices in the path (including the target) not being available (e.g. powered off, cable disconnected). Though less likely, the “Unconnected Request Timed Out” error can also result when there is network congestion and/or one or more of the devices along the path including the target take longer than the time provided by the originator to process the unconnected request. When an “Unconnected Request Timed Out” error persists and it is known through other means that all nodes are available, the originator timeout value should be increased after taking the topology into account.

A default originator timeout value of 5 s is recommended; based on the case of an originator request covering 2 hops (allowing the default 512 ms for each hop and about 4 s for the target).

#### **4.1.6.7 Connection serial number**

The connection serial number shall be a unique 16-bit value selected by the Connection Manager at the originator of the connection. The originator shall make sure that the 16-bit value is unique for the device. There shall be no other significance placed on the number by any other nodes in the connection path. The connection serial numbers shall be unique but do not have to be sequential. For example, an operator interface may have a large number of connections open at the same time, each with a unique number. The same values could be repeated at other operator interface stations. A possible implementation would be to have a connection list which points to the descriptor for each connection, and the connection serial number could be the index into the table.

#### **4.1.6.8 Vendor ID**

The vendor number shall be a unique number assigned to the various vendors of products. Each vendor has a unique number assigned.

NOTE Vendor IDs are assigned by the ODVA, Inc. organization.

#### 4.1.6.9 Originator serial number

The originator serial number shall be a unique 32-bit value that is assigned to a device at the time of manufacture. This value shall be guaranteed to be unique for all devices manufactured by the same vendor. No significance shall be attached to the number. The combination of Vendor ID and Originator Serial Number shall be unique throughout the entire system.

#### 4.1.6.10 Connection number

The connection number shall be a 16-bit value that is assigned by the Connection Manager when a connection is opened. This value allows other nodes to obtain connection data from the Connection Manager. This number shall not be confused with the Connection Serial Number.

#### 4.1.6.11 Connection path size

The connection path size shall be the length of the connection path in 16-bit words. The length of the connection path varies during the connection process, since each node in the connection path removes the current port segment and forwards only the remaining path segments to the next node.

#### 4.1.6.12 Connection path /connection remaining path

The connection path parameter shall contain one or more encoded paths as required by a combination of the service and the value of other parameters within the service data.

The connection path shall be of the form:

|                                              |   |                                               |
|----------------------------------------------|---|-----------------------------------------------|
| [[Electronic Key segment1]                   | } |                                               |
| [Network segment11]...[Network segment1N]    | } | Port Segment Group for 1 <sup>st</sup> router |
| Port segment1]                               | } |                                               |
| [[Electronic Key segment2]                   | } |                                               |
| [Network segment21]...[Network segment2N]    | } | Port Segment Group for 2 <sup>nd</sup> router |
| Port segment2]                               | } |                                               |
| ...                                          |   |                                               |
| [[Electronic Key segmentM]                   | } |                                               |
| [Network segmentM1]...[Network segmentMN]    | } |                                               |
| [[Application Path]                          | } | Port Segment Group for M <sup>th</sup> router |
| [Network segmentO1] ... [Network segmentON]] | } |                                               |
| Port segmentM]                               |   |                                               |
| [Electronic Key segmentP]                    |   |                                               |
| [Network segmentP1]...[Network segmentPN]    | } |                                               |
| Application Path1                            | } |                                               |
| [Application Path2]                          | } | Application Segment Group for target          |
| [Application Path3]                          | } |                                               |
| [Simple Data segment]                        | } |                                               |
| [Network segmentQ1] ... [Network segmentQN]  |   |                                               |

NOTE In the last format above, for the target, the network segment(s) that follow the application path(s) and the data segment are limited to certain safety network segments as defined by IEC 61784-3-2.

All segments listed in brackets [...] are optional.

Network segments are allowed at the specified locations only if allowed by the specific network segment definition.

When a configuration data segment is included in the connection path the configuration data segment shall follow the applications path(s).

The components of a connection path are specified in 4.1.9.1 to 4.1.9.7, and the hierarchy of a connection path is specified in 4.1.9.8.

The Connection Manager object of each intermediate and target node shall process all segments preceding application path(s) that are directed to it. For the last-hop router ( $M^{\text{th}}$  router), the use of network segments that follow the application path shall be directly associated with the object identified by that application path and that object shall define its use.

Depending on the O2T\_connection\_parameters and T2O\_connection\_parameters fields and the presence of a data segment, one or more encoded application paths shall be specified. In general, the application paths are in the order of configuration path, consumption path, and production path. However, a single encoded path can be used when configuration, consumption, and/or production use the same path. See Table 34 for the valid combinations and the implied meaning of the application paths. The application paths are relative to the target node.

Any O2T or T2O application path may be a heartbeat application path, however the following application paths are defined for the Assembly object:

- “20 04 2C ED” is an O2T Listen Only Heartbeat
- “20 04 2C EE” is an O2T Input Only Heartbeat
- “20 04 2C EF” is a T2O Heartbeat

These pre-defined application paths shall be used for heartbeat connections when the target device is reached through a Type 2 router to a target device that does not support the Connection Manager object.

These pre-defined heartbeat application paths may also be used for heartbeat connections when the target device supports the Connection Manager object.

**Table 34 – Encoded application path ordering**

| Network connection parameters |                     | Data segment present | Number of encoded application paths                   |                                                                                |                                                                                               |
|-------------------------------|---------------------|----------------------|-------------------------------------------------------|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| O2T connection type           | T2O connection type |                      | 1                                                     | 2                                                                              | 3                                                                                             |
| NULL                          | NULL                | Yes                  | Path is for configuration                             | First path is for configuration, second path is ignored                        | First path is for configuration, second and third paths are ignored                           |
|                               |                     | No                   | Path is for pinging a device <sup>a</sup>             | Invalid                                                                        | Invalid                                                                                       |
| not NULL                      | NULL                | Yes                  | Path is for configuration and consumption             | First path is for configuration, second path is for consumption                | Invalid                                                                                       |
|                               |                     | No                   | Path is for consumption                               | Invalid                                                                        | Invalid                                                                                       |
| NULL                          | not NULL            | Yes                  | Path is for configuration and production              | First path is for configuration, second path is for production                 | Invalid                                                                                       |
|                               |                     | No                   | Path is for production                                | Invalid                                                                        | Invalid                                                                                       |
| not NULL                      | not NULL            | Yes                  | Path is for configuration, consumption and production | First path is for configuration, second path is for consumption and production | First path is for configuration, second path is for consumption, third path is for production |
|                               |                     | No                   | Path is for consumption and production                | First path is for consumption, second path is for production                   | First path is ignored, second path is for consumption, third path is for production           |

<sup>a</sup> If path is “20 01 24 01” used to ping a device, see 4.1.5.3.2.3 (null Forward\_Open, not matching), invalid otherwise.

Encoded path compression rules are specified in 4.1.9.9.

#### 4.1.6.13 Network connection ID

The Network Connection ID shall be link specific and shall not be related to the connection serial number, which is connection specific and the same over all the links. The fields of the Network Connection ID shall be used to set the screening mechanism for the specified link.

The Network connection ID is either a produced connection ID or a consumed connection ID.

A multicast connection ID shall not be reused until all connections associated with the connection ID have been closed or timed out.

The network connection ID is further specified in the DMPMs corresponding to the different link layers which can be associated with the Type 2 application layer (see Clause 10, Clause 11 and Clause 12).

#### 4.1.6.14 Transport class and trigger

The transport class and trigger specify the type of transport and the production trigger required for the connection. The octet that encode the transport class and trigger shall be of the following form shown in Table 35.

**Table 35 – Transport class, trigger and Is\_Server format**

| TransportClass (4 bits)                                              | TriggerMode (3 bits)                                 | Is_Server (1 bit)                  |
|----------------------------------------------------------------------|------------------------------------------------------|------------------------------------|
| NULL = 0<br>DUPLICATE_DETECT = 1<br>ACKNOWLEDGED = 2<br>VERIFIED = 3 | CYCLIC = 0<br>CHANGE_OF_STATE = 1<br>APPLICATION = 2 | IS_NOT_SERVER = 0<br>IS_SERVER = 1 |

The least significant 4 bits, `class`, shall determine which transport type is specified and shall be in the range 0 to 6. The `class` values 7 through 15 shall be reserved. Bits 4 to 6, called `trigger`, shall determine what event causes the production of data on the connection and shall be in the range 0 to 2. The `trigger` values 3 through 7 shall be reserved. For transports that need to specify the target as a server, the `is_server` field shall be 1; otherwise, it shall be 0. For transport classes 0 and 1, the `is_server` field shall be ignored. Bit 7 shall be the `is_server` field;

See IEC 61158-5-2 for details on the behavior for the different combinations of `is_server`, `class`, and `trigger`, `CM_RPI`, and `CM_API`.

#### 4.1.7 MR headers

##### 4.1.7.1 MR request packets

All request packets delivered to the Message Router, either from a connection or from the UCMM, shall have a header of the form shown in Table 36.

**Table 36 – MR\_Request\_Header format**

| Parameter | Format       |
|-----------|--------------|
| Service   | USINT        |
| path_size | USINT        |
| path      | Padded EPATH |

The first octet, `service`, shall be directly delivered to the application object and shall be in the range 1 through 127. The `path_size` shall determine the number of 16-bit words in the `path` field.

The `path` field shall contain addressing information for the packet (application path) and other information, and shall be padded.

The `path` field shall be of the form:

[Electronic Key segment]

Application Path

The Electronic Key segment is conditional. If the request is being sent across a connection the Electronic Key segment is not allowed, otherwise it is optional. If the Electronic Key segment is present the target shall evaluate the key. See 4.1.9.4.2 for the definition of Electronic Key segment and see 4.1.9.8 for the definition of the Application Path format.

EXAMPLE

34 04 42 42 0C 00 01 00 81 01      Electronic Key segment indicating the target device shall be compatible with a Vendor 0x4242, Device Type 0x000C, Product Code 0x0001, Major Revision 0x01, Minor Revision 0x01 device

20 04 24 01 30 03      Application path that specifies Instance 1 of the Assembly Object, Attribute 3.

#### 4.1.7.2 MR response packets

Response packets from the Message Router shall have a header of the form shown in Table 37.

**Table 37 – MR\_Response\_Header format**

| Parameter            | Format |
|----------------------|--------|
| Service              | USINT  |
| Reserved             | USINT  |
| general_status       | USINT  |
| Extended_status_size | USINT  |
| Extended_status[]    | WORD   |

The service field shall be the value of the MR\_Request\_Header.service field with its most significant bit set (i.e. plus 0x80). If MR\_Request\_Header.service equals 0x05, the MR\_Response\_Header.service shall equal 0x85. The general\_status field shall be filled using the Status Code parameter of the service response. Likewise the extended\_status, extended\_status\_size, and response shall be filled using the Extended Status and other response parameters from the service response, respectively. If extended\_status\_size is 0, there is no extended\_status. The response\_size shall not be explicitly included in the MR\_Response\_Header; it shall be implied by number of octets in the MR\_Response\_Header.

#### 4.1.8 OM\_Service\_PDU

##### 4.1.8.1 General syntax

##### 4.1.8.1.1 Get\_Attributes\_All

The Get\_Attributes\_All\_RequestPDU body is empty.

The Get\_Attributes\_All\_ResponsePDU body shall be as specified in Table 38.

**Table 38 – Structure of Get\_Attributes\_All\_ResponsePDU body**

| Name           | Data type                       |
|----------------|---------------------------------|
| Attribute Data | Object/class attribute-specific |

##### 4.1.8.1.2 Set\_Attributes\_All

The Set\_Attributes\_All\_RequestPDU body shall be as specified in Table 39.

**Table 39 – Structure of Set\_Attributes\_All\_RequestPDU body**

| Name           | Data type                       |
|----------------|---------------------------------|
| Attribute Data | Object/class attribute-specific |

The Set\_Attributes\_All\_ResponsePDU body is empty.

#### 4.1.8.1.3 Get\_Attribute\_List

The Get\_Attribute\_List\_RequestPDU body shall be as specified in Table 40.

**Table 40 – Structure of Get\_Attribute\_List\_RequestPDU body**

| Name            | Data type     |
|-----------------|---------------|
| Attribute Count | UINT          |
| Attribute List  | Array of UINT |

The Get\_Attribute\_List\_ResponsePDU body shall be as specified in Table 41.

**Table 41 – Structure of Get\_Attribute\_List\_ResponsePDU body**

| Name                          | Data type                        | Semantics of values                                                  |
|-------------------------------|----------------------------------|----------------------------------------------------------------------|
| Attribute Count               | UINT                             |                                                                      |
| Attribute Response Structures | Array of STRUCT                  |                                                                      |
| Attribute ID                  | UINT                             |                                                                      |
| Attribute Status              | USINT                            | See 4.1.11.2.1                                                       |
|                               | OCTET                            | Reserved (shall be set to 0)                                         |
| Attribute Data                | Object/class attribute- specific | Shall be omitted if the Attribute status value is not 0x00 (Success) |

#### 4.1.8.1.4 Set\_Attribute\_List

The Set\_Attribute\_List\_RequestPDU body shall be as specified in Table 42.

**Table 42 – Structure of Set\_Attribute\_List\_RequestPDU body**

| Name                         | Data type                        |
|------------------------------|----------------------------------|
| Attribute Count              | UINT                             |
| Attribute Request Structures | Array of STRUCT                  |
| Attribute ID                 | UINT                             |
| Attribute Value              | Object/class attribute- specific |

The Set\_Attribute\_List\_ResponsePDU body shall be as specified in Table 43.

**Table 43 – Structure of Set\_Attribute\_List\_ResponsePDU body**

| Name                          | Data type                        | Semantics of values                                             |
|-------------------------------|----------------------------------|-----------------------------------------------------------------|
| Attribute Count               | UINT                             |                                                                 |
| Attribute Response Structures | Array of STRUCT                  |                                                                 |
| Attribute ID                  | UINT                             |                                                                 |
| Attribute Status              | USINT                            | See 4.1.11.2.1                                                  |
|                               | OCTET                            | Reserved (shall be set to 0)                                    |
| Attribute Data                | Object/class attribute- specific | If defined by the object and Attribute Status is 0x00 (Success) |

**4.1.8.1.5 Reset**

The Reset\_RequestPDU body shall be as specified in Table 44, if the **optional** “Object Specific Data” service parameter of the Reset request is specified. Else it shall be empty.

**Table 44 – Structure of Reset\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Reset\_ResponsePDU body shall be as specified in Table 45, if the **optional** “Object Specific Data” service parameter of the Reset response is specified. Else it shall be empty.

**Table 45 – Structure of Reset\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

**4.1.8.1.6 Start**

The Start\_RequestPDU body shall be as specified in Table 46, if the **optional** “Object Specific Data” service parameter of the Start request is specified. Else it shall be empty.

**Table 46 – Structure of Start\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Start\_ResponsePDU body shall be as specified in Table 47, if the **optional** “Object Specific Data” service parameter of the Start response is specified. Else it shall be empty.

**Table 47 – Structure of Start\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

#### 4.1.8.1.7 Stop

The Stop\_RequestPDU body shall be as specified in Table 48, if the **optional** “Object Specific Data” service parameter of the Stop request is specified. Else it shall be empty.

**Table 48 – Structure of Stop\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Stop\_ResponsePDU body shall be as specified in Table 49, if the **optional** “Object Specific Data” service parameter of the Stop response is specified. Else it shall be empty.

**Table 49 – Structure of Stop\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

#### 4.1.8.1.8 Create

The Create\_RequestPDU body shall be as specified in Table 50, if the **optional** “Object Specific Data” service parameter of the Create request is specified. Else it shall be empty.

**Table 50 – Structure of Create\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Create\_ResponsePDU body shall be as specified in Table 51, if the **optional** “Object Specific Data” service parameter of the Create response is specified. Else it shall be empty.

**Table 51 – Structure of Create\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Instance ID          | UINT                           |
| Object Specific Data | Object/class service- specific |

#### 4.1.8.1.9 Delete

The Delete\_RequestPDU body shall be as specified in Table 52, if the **optional** “Object Specific Data” service parameter of the Delete request is specified. Else it shall be empty.

**Table 52 – Structure of Delete\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Delete\_ResponsePDU body shall be as specified in Table 53, if the **optional** “Object Specific Data” service parameter of the Delete response is specified. Else it shall be empty.

**Table 53 – Structure of Delete\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

**4.1.8.1.10 Get\_Attribute\_Single**

The Get\_Attribute\_Single\_RequestPDU body is empty, except as noted below.

Some Type network communication profiles support link specific explicit message formats (i.e. they do not support the Message Router request format), which do not enable the Attribute ID to be specified in the Path parameter. For those formats, the Attribute ID is limited to a USINT and is transmitted as Get\_Attribute\_Single\_ResponsePDU body.

The Get\_Attribute\_Single\_ResponsePDU body shall be as specified in Table 54.

**Table 54 – Structure of Get\_Attribute\_Single\_ResponsePDU body**

| Name           | Data type                       |
|----------------|---------------------------------|
| Attribute Data | Object/class attribute-specific |

**4.1.8.1.11 Set\_Attribute single**

The Set\_Attribute\_Single\_RequestPDU body shall be as specified in Table 55.

**Table 55 – Structure of Set\_Attribute\_Single\_RequestPDU body**

| Name           | Data type                       |
|----------------|---------------------------------|
| Attribute Data | Object/class attribute-specific |

Some Type network communication profiles support link specific explicit message formats (i.e. they do not support the Message Router request format), which do not enable the Attribute ID to be specified in the Path parameter. For those formats, the Attribute ID is limited to a USINT and is transmitted as the first parameter (octet) of the Set\_Attribute\_Single\_RequestPDU body.

The Set\_Attribute\_Single\_ResponsePDU body shall be as specified in Table 56, if the **optional** “Object Specific Data” service parameter of the Set\_Attribute\_Single response is specified. Else it shall be empty.

**Table 56 – Structure of Set\_Attribute\_Single\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

**4.1.8.1.12 Find\_Next\_Object\_Instance**

The Find\_Next\_Object\_Instance\_RequestPDU body shall be as specified in Table 57.

**Table 57 – Structure of Find\_Next\_Object\_Instance\_RequestPDU body**

| Name                    | Data type |
|-------------------------|-----------|
| Maximum Returned Values | USINT     |

The Find\_Next\_Object\_Instance\_ResponsePDU body shall be as specified in Table 58.

**Table 58 – Structure of Find\_Next\_Object\_Instance\_ResponsePDU body**

| Name                   | Data type     |
|------------------------|---------------|
| Number Of List Members | USINT         |
| Instance ID List       | Array of UINT |

#### 4.1.8.1.13 NOP

The NOP\_RequestPDU body is empty.

The NOP\_ResponsePDU body is empty.

#### 4.1.8.1.14 Apply\_Attributes

The Apply\_Attributes\_RequestPDU body shall be as specified in Table 59, if the **optional** “Object Specific Data” service parameter of the Apply\_Attributes request is specified. Else it shall be empty.

**Table 59 – Structure of Apply\_Attributes\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Apply\_Attributes\_ResponsePDU body shall be as specified in Table 60, if the **optional** “Object Specific Data” service parameter of the Apply\_Attributes response is specified. Else it shall be empty.

**Table 60 – Structure of Apply\_Attributes\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

#### 4.1.8.1.15 Save

The Save\_RequestPDU body shall be as specified in Table 61, if the **optional** “Object Specific Data” service parameter of the Save request is specified. Else it shall be empty.

**Table 61 – Structure of Save\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Save\_ResponsePDU body shall be as specified in Table 62, if the **optional** “Object Specific Data” service parameter of the Save response is specified. Else it shall be empty.

**Table 62 – Structure of Save\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

**4.1.8.1.16 Restore**

The Restore\_RequestPDU body shall be as specified in Table 63, if the **optional** “Object Specific Data” service parameter of the Restore request is specified. Else it shall be empty.

**Table 63 – Structure of Restore\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Restore\_ResponsePDU body shall be as specified in Table 64, if the **optional** “Object Specific Data” service parameter of the Restore response is specified. Else it shall be empty.

**Table 64 – Structure of Restore\_ResponsePDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

**4.1.8.1.17 Get\_Member**

The Get\_Member\_RequestPDU body is empty.

The Get\_Member\_ResponsePDU body shall be as specified in Table 65.

**Table 65 – Structure of Get\_Member\_ResponsePDU body**

| Name        | Data type                  |
|-------------|----------------------------|
| Member ID   | See 4.1.8.1.21 for details |
| Member Data |                            |

**4.1.8.1.18 Set\_Member**

The Set\_Member\_RequestPDU body shall be as specified in Table 66.

**Table 66 – Structure of Set\_Member\_RequestPDU body**

| Name        | Data type                  |
|-------------|----------------------------|
| Member Data | See 4.1.8.1.21 for details |

The Set\_Member\_ResponsePDU body shall be as specified in Table 67, if the **optional** “Member Data” service parameter of the Set\_Member response is specified. Else it shall be empty.

**Table 67 – Structure of Set\_Member\_ResponsePDU body**

| Name        | Data type                  |
|-------------|----------------------------|
| Member ID   | See 4.1.8.1.21 for details |
| Member Data |                            |

**4.1.8.1.19 Insert\_Member**

The Insert\_Member\_RequestPDU body shall be as specified in Table 68, if the **optional** “Member Data” service parameter of the Insert\_Member request is specified. Else it shall be empty.

**Table 68 – Structure of Insert\_Member\_RequestPDU body**

| Name        | Data type                  |
|-------------|----------------------------|
| Member Data | See 4.1.8.1.21 for details |

The Insert\_Member\_ResponsePDU body shall be as specified in Table 69. If the **optional** “Member Data” service parameter of the Insert\_Member response is not specified, this field shall be empty.

**Table 69 – Structure of Insert\_Member\_ResponsePDU body**

| Name        | Data type                  |
|-------------|----------------------------|
| Member ID   | See 4.1.8.1.21 for details |
| Member Data |                            |

**4.1.8.1.20 Remove\_Member**

The Remove\_Member\_RequestPDU body is empty.

The Remove\_Member\_ResponsePDU body shall be as specified in Table 70.

**Table 70 – Structure of Remove\_Member\_ResponsePDU body**

| Name        | Data type                  |
|-------------|----------------------------|
| Member ID   | See 4.1.8.1.21 for details |
| Member Data |                            |

**4.1.8.1.21 Common syntax elements for the \_Member services****4.1.8.1.21.1 General**

Attributes of object classes may consist of arrays of basic data types or structures. To manipulate members of an array, the following member services use the common syntax elements defined in 4.1.8.1.21:

- Get\_Member;
- Set\_Member;
- Insert\_Member;
- Remove\_Member.

The first member in an array is specified by a Member ID value of one (1).

#### 4.1.8.1.21.2 Member ID/EX description

Two message formats are defined for member services: basic and extended. Within the extended format, there exist up to 255 protocol options. The message format is selected by the most significant bit of the Member ID.

- When a subnet does not support Attribute and Member level addressing, the Member ID/EX parameter is sent as a 16 bit (WORD) parameter within the service data.
- If the subnet does support Attribute and Member level addressing, the Member ID/EX parameter is sent within the Logical Segment and can be either 8, 16, or 32 bits (SWORD, WORD, DWORD).

Figure 6 defines the bits contained within the Member ID/EX field when sent as a WORD.

| Bits                  |                        |   |   |   |   |   |   |
|-----------------------|------------------------|---|---|---|---|---|---|
| 7                     | 6                      | 5 | 4 | 3 | 2 | 1 | 0 |
| Member ID Bits 0 to 7 |                        |   |   |   |   |   |   |
| EX                    | Member ID Bits 8 to 14 |   |   |   |   |   |   |

**Figure 6 – Member ID/EX description (WORD)**

The lower bits (0 to 14 for a WORD) of the Member ID/EX field identify the member to be manipulated. The highest bit EX (15 for a WORD), selects either the basic (EX=0) or extended (EX=1) message format.

#### 4.1.8.1.21.3 Member request – basic format

Table 71 specifies the common structure basic of the \_RequestPDU body for Member services using the basic format.

**Table 71 – Common structure of \_Member\_RequestPDU body (basic format)**

| Name                       | Data type       | Description of parameter                                                                                                                                                                |
|----------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Attribute ID <sup>a</sup>  | USINT           | Identifies the attribute the member is to be serviced.                                                                                                                                  |
| Member ID/ EX <sup>b</sup> | WORD            | The lower 15 bits identifies the Member ID of the member to be serviced. If the value is zero (0) the service used determines the behavior.<br>Extended Protocol Option EX = 0 (bit 15) |
| Member Data (conditional)  | Member specific | This field is required by some services, see individual service descriptions.                                                                                                           |

<sup>a</sup> This parameter shall be present when the target of the request is on a subnet which does not support explicit message addressing to the attribute level. If the subnet does support attribute level addressing then this parameter shall not be present. In the latter case, the data type of the Attribute ID can be either USINT, UINT, or UDINT.

<sup>b</sup> This parameter shall be present when the target of the request is on a subnet which does not support explicit message addressing to the member or attribute level. If the subnet does support attribute and member level addressing then this parameter shall not be present. In this latter case, the data type of the Member ID/EX can be either SWORD, WORD, or DWORD with the most significant bit being the Extended Protocol Option parameter and the remaining bits being the Member ID.

Table 72 specifies the common structure basic of the \_ResponsePDU body for Member services using the basic format.

**Table 72 – Common structure of \_Member\_ResponsePDU body (basic format)**

| Name                      | Data type       | Description of parameter                                                                                                                                                        |
|---------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Member ID (conditional)   | UINT            | The Member ID of the member serviced. This field is required by some services, see individual service descriptions. This field is required if the Member Data field is present. |
| Member Data (conditional) | Member specific | This field is required by some services, see individual service descriptions.                                                                                                   |

**4.1.8.1.21.4 Member request – extended format (general)**

If the Extended protocol bit [EX] of the Member ID/EX field is set to a one [1], the octet following the Member ID/EX defines the remainder of the protocol.

Ranges of values for the Extended Protocol ID are divided into three categories (open, vendor specific, reserved), as specified in 4.1.10.1.1.

Table 73 specifies the common structure basic of the \_RequestPDU body for Member services using the extended format.

**Table 73 – Common structure of \_Member\_RequestPDU body (extended format)**

| Name                       | Data type         | Description of parameter                                                                                                                                                                |
|----------------------------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Attribute ID <sup>a</sup>  | USINT             | Identifies the attribute the member is to be serviced.                                                                                                                                  |
| Member ID/ EX <sup>b</sup> | WORD              | The lower 15 bits identifies the Member ID of the member to be serviced. If the value is zero (0) the service used determines the behavior.<br>Extended Protocol Option EX = 1 (bit 15) |
| Extended Protocol ID       | USINT             | Selects the extended protocol used for the remainder of message. See Table 75.                                                                                                          |
| Additional data            | Protocol specific | Additional request data is defined by the extended protocol.                                                                                                                            |

<sup>a</sup> This parameter shall be present when the target of the request is on a subnet which does not support explicit message addressing to the attribute level. If the subnet does support attribute level addressing then this parameter shall not be present. In the latter case, the data type of the Attribute ID can be either USINT, UINT, or UDINT.

<sup>b</sup> This parameter shall be present when the target of the request is on a subnet which does not support explicit message addressing to the member or attribute level. If the subnet does support attribute and member level addressing then this parameter shall not be present. In this latter case, the data type of the Member ID/EX can be either SWORD, WORD, or DWORD with the most significant bit being the Extended Protocol Option parameter and the remaining bits being the Member ID.

Table 74 specifies the common structure basic of the \_ResponsePDU body for Member services using the extended format.

**Table 74 – Common structure of \_Member\_ResponsePDU body (extended format)**

| Name          | Data type         | Description of parameter                                  |
|---------------|-------------------|-----------------------------------------------------------|
| Response data | Protocol specific | Response data is defined by the extended protocol option. |

Table 75 lists the values currently defined for the Extended Protocol ID.

**Table 75 – Extended Protocol ID**

| Value                                                                       | Description                    |
|-----------------------------------------------------------------------------|--------------------------------|
| 0x00                                                                        | Open                           |
| 0x01                                                                        | Multiple Sequential Members    |
| 0x02                                                                        | International String Selection |
| 0x03 to 0x63                                                                | Open                           |
| 0x64 to 0xC7                                                                | Vendor specific                |
| 0xC8 to 0xFF                                                                | Reserved                       |
| NOTE Reserved ranges can be further defined by the ODVA, Inc. organization. |                                |

#### 4.1.8.1.21.5 Member request – extended format (Multiple Sequential Members)

When the Extended Protocol ID is set to Multiple Sequential Members [1], the structure of the \_RequestPDU body is specified in Table 76.

**Table 76 – Structure of \_Member\_RequestPDU body (Multiple Sequential Members)**

| Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Data type       | Description of parameter                                                                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Attribute ID <sup>a</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | USINT           | Identifies the attribute the member is to be serviced.                                                                                                                                  |
| Member ID/ EX <sup>b</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | WORD            | The lower 15 bits identifies the Member ID of the member to be serviced. If the value is zero (0) the service used determines the behavior.<br>Extended Protocol Option EX = 1 (bit 15) |
| Extended Protocol ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | USINT           | Selects the Multiple Sequential Members protocol option (value = 1).                                                                                                                    |
| Number of Members                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT            | Number of members to be serviced.                                                                                                                                                       |
| Member Data (conditional)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Member specific | Multiple sequential Member Data.<br>This field is required by some services, see individual service descriptions.                                                                       |
| <sup>a</sup> This parameter shall be present when the target of the request is on a subnet which does not support explicit message addressing to the attribute level. If the subnet does support attribute level addressing then this parameter shall not be present. In the latter case, the data type of the Attribute ID can be either USINT, UINT, or UDINT.<br><sup>b</sup> This parameter shall be present when the target of the request is on a subnet which does not support explicit message addressing to the member or attribute level. If the subnet does support attribute and member level addressing then this parameter shall not be present. In this latter case, the data type of the Member ID/EX can be either SWORD, WORD, or DWORD with the most significant bit being the Extended Protocol Option parameter and the remaining bits being the Member ID. |                 |                                                                                                                                                                                         |

The structure of the \_ResponsePDU body is specified in Table 77.

**Table 77 – Structure of \_Member\_ResponsePDU body (Multiple Sequential Members)**

| Name                      | Data type       | Description of parameter                                                                                                                                                                |
|---------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Number of Members         | UINT            | Number of members serviced. This field is required                                                                                                                                      |
| Member ID (conditional)   | UINT            | Member ID of the first member serviced.<br>This field is required by some services, see individual service descriptions.<br>This field is required if the Member Data field is present. |
| Member Data (conditional) | Member specific | Multiple sequential Member Data.<br>This field is required by some services, see individual service descriptions.                                                                       |

#### 4.1.8.1.21.6 Member request – extended format (International String Selection)

This service returns a single international character string from an attribute, which has a data type of STRINGI. This service can request a specific language or default to the current active language as specified by the Identity object.

The International String Selection protocol is only valid with the Get\_Member service.

The structure of the \_RequestPDU body is specified in Table 78.

**Table 78 – Structure of \_Member\_RequestPDU body (International String Selection)**

| Name                 | Data type | Description of parameter                                                                                                                                                                                                                                              |
|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Attribute ID         | USINT     | Identifies the attribute containing the international string.                                                                                                                                                                                                         |
| Member ID/ EX        | WORD      | The lower 15 bits identifies the language to retrieve. This value is the member ID (array index, starting at one) of the language within the Supported Language List attribute (Attribute ID 12) of the Identity object.<br>Extended Protocol Option EX = 1 (bit 15). |
| Extended Protocol ID | USINT     | Selects the International String Selection protocol option (value = 2).                                                                                                                                                                                               |

The structure of the \_ResponsePDU body is specified in Table 79.

**Table 79 – Structure of \_Member\_ResponsePDU body (International String Selection)**

| Name                 | Data type    | Description of parameter                                                                                                                                                 |
|----------------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| International String | STRUCT of    | The character string in the requested language. This parameter follows the definition of a single international character string within the STRINGI data type (see 4.2). |
|                      | USINT        | The language1 field from the STRINGI data type.                                                                                                                          |
|                      | USINT        | The language2 field from the STRINGI data type.                                                                                                                          |
|                      | USINT        | The language3 field from the STRINGI data type.                                                                                                                          |
|                      | USINT        | The structure of the character string, limited to the values 0xD0, 0xD5, 0xD9 and 0xDA.                                                                                  |
|                      | UINT         | Character set of the international string.                                                                                                                               |
|                      | OCTET_STRING | International string                                                                                                                                                     |

#### 4.1.8.1.22 Group\_Sync

The Group\_Sync\_RequestPDU body shall be as specified in Table 80.

**Table 80 – Structure of Group\_Sync\_RequestPDU body**

| Name                 | Data type                      |
|----------------------|--------------------------------|
| Object Specific Data | Object/class service- specific |

The Group\_Sync\_ResponsePDU body shall be as specified in Table 81.

**Table 81 – Structure of Group\_Sync\_ResponsePDU body**

| Name                 | Data type                      | Semantics of values                                                                                       |
|----------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------|
| IsSynchronized       | BOOL                           | Indicates if object is synchronized to the PTP Time Master<br>0 = Not synchronized<br>1 = Is synchronized |
| Object Specific Data | Object/class service- specific | Object/class service- specific                                                                            |

**4.1.8.1.23 Multiple\_Service\_Packet**

The Multiple\_Service\_Packet\_RequestPDU body shall be as specified in Table 82.

**Table 82 – Structure of Multiple\_Service\_Packet\_RequestPDU body**

| Name               | Data type       | Semantics of values                                                                                                                                                                                                                          |
|--------------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Number of Services | UINT            | Number of embedded services in the Service List.                                                                                                                                                                                             |
| Service Offsets    | ARRAY of UINT   | Array of octet offsets to the start of each embedded service in the Service List.<br><br>Each octet offset is relative to the Number of Services parameter. Therefore the first element in this array shall be (2 + Number of Services x 2). |
| Service List       | Array of STRUCT | Array of service request structures.<br><br>The format of the service request follows the Message Router Request header.                                                                                                                     |
|                    | USINT           | Service code of the request.                                                                                                                                                                                                                 |
|                    | USINT           | The number of 16 bit words in the Request_Path field (next element).                                                                                                                                                                         |
|                    | Padded EPATH    | This is an array of octets whose contents convey the path of the request (Class ID, Instance ID, etc.) for this transaction.                                                                                                                 |
|                    | ARRAY of octet  | Service specific data to be delivered in the Explicit Messaging Request. If no additional data is to be sent with the Explicit Messaging Request, then this array will be empty.                                                             |

The Multiple\_Service\_Packet\_ResponsePDU body shall be as specified in Table 83.

**Table 83 – Structure of Multiple\_Service\_Packet\_ResponsePDU body**

| Name                | Data type       | Semantics of values                                                                                                                                                                                                                                                     |
|---------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Number of Responses | UINT            | Number of embedded service responses in the Response List.                                                                                                                                                                                                              |
| Response Offsets    | ARRAY of UINT   | Array of octet offsets to the start of each embedded service response in the Response List.<br><br>Each octet offset is relative to the Number of Responses parameter. Therefore the first element in this array shall be $(2 + \text{Number of Responses} \times 2)$ . |
| Response List       | Array of STRUCT | Array of service response structures.<br><br>The format of the service response follows the Message Router Response header.                                                                                                                                             |
|                     | USINT           | Response service code.                                                                                                                                                                                                                                                  |
|                     | USINT           | Shall be zero.                                                                                                                                                                                                                                                          |
|                     | USINT           | One of the General Status codes.                                                                                                                                                                                                                                        |
|                     | USINT           | Number of 16 bit words in Extended Status array.                                                                                                                                                                                                                        |
|                     | ARRAY of UINT   | Extended status.                                                                                                                                                                                                                                                        |
|                     | ARRAY of octet  | Response data from request or additional error data if General Status indicated an error.                                                                                                                                                                               |

#### 4.1.8.1.24 Get\_Connection\_Point\_Member\_List

The Get\_Connection\_Point\_Member\_List\_RequestPDU body is empty.

The Get\_Connection\_Point\_Member\_List\_ResponsePDU body shall be as specified in Table 84.

**Table 84 – Structure of Get\_Connection\_Point\_Member\_List\_ResponsePDU body**

| Name             | Data type       | Semantics of values                                                                                                                             |
|------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Item Count       | UINT            | Number of STRUCTs returned                                                                                                                      |
| Members          | Array of STRUCT | A list of EPATHs of each member of the Connection Point structure, or the EPATH to the full Parameter Object Instance that refers to the member |
| Member Data Size | UINT            | Size of member data (in bits)                                                                                                                   |
| Member Path Size | UINT            | Size of member path (in octets)                                                                                                                 |
| Member Path      | Packed EPATH    | EPATH of the member or EPATH of the full Parameter object instance that refers to the member                                                    |

#### 4.1.8.2 Identity object specific syntax elements

##### 4.1.8.2.1 Attributes

##### 4.1.8.2.1.1 Class attributes

The format of the Identity object class attributes shall be as specified in Table 85.

**Table 85 – Identity object class attributes**

| Attribute ID    | Name                                                         | Data type | Semantics of values                                                                                                                                                                                            |
|-----------------|--------------------------------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1               | Revision                                                     | UINT      | Revision of this object.<br>Value = 1 or 2<br>The value shall be 2 if attributes greater than #10 are included in the Get_Attributes_All response (recommended when applicable) or attribute #30 is supported. |
| 2               | Max Instance                                                 | UINT      |                                                                                                                                                                                                                |
| 3 – 7, 200, 201 | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                                                                                                                                                                                                |

**4.1.8.2.1.2 Instance attributes**

The format of the Identity object instance attributes is as specified in Table 86.

**Table 86 – Identity object instance attributes**

| Attribute ID | Name                            | Data type    | Semantics of values                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|---------------------------------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1            | Vendor ID                       | UINT         | The value zero shall not be used.<br>A list of the Vendor ID ranges can be found in Table 87.                                                                                                                                                                                                                                                                                                |
| 2            | Device Type                     | UINT         | A list of the Device Type ranges can be found in 4.1.10.3.                                                                                                                                                                                                                                                                                                                                   |
| 3            | Product Code                    | UINT         | The value zero is not valid.                                                                                                                                                                                                                                                                                                                                                                 |
| 4            | Revision                        | STRUCT of    | Revision of the hardware the Identity object represents.<br>The value zero is not valid for either the Major or Minor Revision fields of a compliant product. If the Device Type of the instance is Embedded Component, the Major Revision and/or Minor Revision may be zero.                                                                                                                |
|              | Major Revision                  | USINT        | The Major Revision attribute shall be limited to values between 1 and 127 (7 bits). The eighth bit is reserved and shall be zero.                                                                                                                                                                                                                                                            |
|              | Minor Revision                  | USINT        | 1 to 255                                                                                                                                                                                                                                                                                                                                                                                     |
| 5            | Status                          | WORD         | Bit definitions are described in Table 88.                                                                                                                                                                                                                                                                                                                                                   |
| 6            | Serial Number                   | UDINT        | The value zero is not a valid serial number and shall not be assigned to a product nor reported by a compliant device. Reception of the value zero when accessing this attribute indicates the target is not a Type 2 device and does not contain a serial number.                                                                                                                           |
| 7            | Product Name                    | SHORT_STRING | The maximum number of 8-bit characters in this string shall be 32.                                                                                                                                                                                                                                                                                                                           |
| 8            | State                           | USINT        | Present state of the device as represented by the state transition diagram:<br><br>0 = Nonexistent<br>1 = Device Self Testing<br>2 = Standby<br>3 = Operational<br>4 = Major Recoverable Fault<br>5 = Major Unrecoverable Fault<br>6 to 254 = Reserved<br>255 = Default value<br><br>The default value shall be used in the Get_Attributes_All response if the attribute is not implemented. |
| 9            | Configuration Consistency Value | UINT         | Contents identify configuration of device                                                                                                                                                                                                                                                                                                                                                    |

| Attribute ID | Name                        | Data type             | Semantics of values                                                                                                                                                                            |
|--------------|-----------------------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10           | Heartbeat Interval          | USINT                 | The nominal interval between heartbeat messages in seconds.<br>The default value is 0. Zero disables transmission of the heartbeat message.                                                    |
| 11           | Active Language             | STRUCT of             | Currently active language for the device (based on ISO 639-2/T).                                                                                                                               |
|              | Language1                   | USINT                 | The language1 field from the STRINGI data type.                                                                                                                                                |
|              | Language2                   | USINT                 | The language2 field from the STRINGI data type.                                                                                                                                                |
|              | Language3                   | USINT                 | The language3 field from the STRINGI data type.                                                                                                                                                |
| 12           | Supported Language List     | ARRAY of<br>STRUCT of | List of languages supported by character strings of data type STRINGI within the device.<br>The array index of the language is used within the International String Selection member protocol. |
|              | Language1                   | USINT                 | The language1 field from the STRINGI data type.                                                                                                                                                |
|              | Language2                   | USINT                 | The language2 field from the STRINGI data type.                                                                                                                                                |
|              | Language3                   | USINT                 | The language3 field from the STRINGI data type.                                                                                                                                                |
| 13           | International Product Name  | STRINGI               | Names of the product in various languages                                                                                                                                                      |
| 14           | Semaphore                   | STRUCT of             | Access Synchronization Semaphore                                                                                                                                                               |
|              | Vendor Number               | UINT                  | Client electronic key                                                                                                                                                                          |
|              | Client Serial Number        | UDINT                 | Default: All zero values                                                                                                                                                                       |
|              | Semaphore Timer             | ITIME                 | Timer valid range 100 ms thru 32 767 ms                                                                                                                                                        |
| 15           | Assigned_Name               | STRINGI               | User assigned name                                                                                                                                                                             |
| 16           | Assigned_Description        | STRINGI               | User assigned description                                                                                                                                                                      |
| 17           | Geographic_Location         | STRINGI               | User assigned location                                                                                                                                                                         |
| 18           | Type15 Identity Info        |                       | Reserved for Type 15 devices.                                                                                                                                                                  |
| 19           | Protection Mode             | WORD                  | Bit definitions are described in Table 90.                                                                                                                                                     |
| 20           | Uptime                      | UDINT                 | Number of seconds                                                                                                                                                                              |
| 21           | Catalog Number              | SHORT STRING          | Textual catalog or model number                                                                                                                                                                |
| 22           | Manufacture Date            | DATE                  | Date the product was manufactured                                                                                                                                                              |
| 23           | Type 9 Identity Info        |                       | Reserved for Type 9 devices                                                                                                                                                                    |
| 24           | Type 9 Status               |                       |                                                                                                                                                                                                |
| 25           | Implementation Profiles     | STRUCT of             | Implementation profiles supported                                                                                                                                                              |
|              |                             | UINT                  | Number of implementation profiles supported                                                                                                                                                    |
|              |                             | ARRAY of<br>STRUCT of |                                                                                                                                                                                                |
|              |                             | UINT                  | Profile number                                                                                                                                                                                 |
|              |                             | DWORD                 | Features supported (see Table 91)                                                                                                                                                              |
| 26           | IEC 61131-9 Identity Info 1 |                       | Reserved for IEC 61131-9 devices                                                                                                                                                               |
| 27           | IEC 61131-9 Identity Info 2 |                       |                                                                                                                                                                                                |
| 28           | IEC 61131-9 Identity Info 3 |                       |                                                                                                                                                                                                |
| 29           | IEC 61131-9 Identity Info 4 |                       |                                                                                                                                                                                                |

| Attribute ID | Name                         | Data type                                               | Semantics of values                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30           | Supported Language List 2    | STRUCT of                                               | List of languages supported by character strings of data type STRINGI within the device. The array index of the language is used within the International String Selection member protocol.                                                                                                                                                              |
|              |                              | UINT                                                    | Number of languages in the list                                                                                                                                                                                                                                                                                                                          |
|              |                              | ARRAY of<br>STRUCT of                                   | List of languages                                                                                                                                                                                                                                                                                                                                        |
|              | Language1                    | USINT                                                   | The language1 field from the STRINGI data type.                                                                                                                                                                                                                                                                                                          |
|              | Language2                    | USINT                                                   | The language2 field from the STRINGI data type.                                                                                                                                                                                                                                                                                                          |
|              | Language3                    | USINT                                                   | The language3 field from the STRINGI data type.                                                                                                                                                                                                                                                                                                          |
| 31           | Vendor Name                  | STRING                                                  | Name of the company that manufactured the device                                                                                                                                                                                                                                                                                                         |
| 32           | Vendor URI                   | STRING                                                  | The maximum length is 246 characters.<br><br>If this attribute is not supported, a client or gateway shall report a value of "www.odva.org/vendor_id_NNNNN", where NNNNN is the value of the Vendor ID attribute in decimal with leading zeroes.<br><br>EXAMPLE - The Vendor URI for a device from Vendor ID 24 would be "www.odva.org/vendor_id_00024". |
| 33           | Configuration Counter        | DINT                                                    | The Configuration Counter shall use an incrementing count mechanism and wrap around from 0xFFFFFFFF to 0x00000000.                                                                                                                                                                                                                                       |
| 34           | Configuration Date           | STIME                                                   | Date and time of last change to Configuration.                                                                                                                                                                                                                                                                                                           |
| 35           | Manufacture Serial Number    | SHORT_STRING                                            | Alphanumeric character sequence used to identify the asset.                                                                                                                                                                                                                                                                                              |
| 36           | Device Manual                | STRUCT of                                               | Points to a manual for the device                                                                                                                                                                                                                                                                                                                        |
|              | Device Manual Data Type      | USINT                                                   | Enumeration indicating the data type for the next field:<br><br>0xDC = Padded EPATH<br>0xD0 = STRING                                                                                                                                                                                                                                                     |
|              | Pointer to Device Manual     |                                                         | Data type is defined in Device Manual Data Type member                                                                                                                                                                                                                                                                                                   |
| 37           | Hardware Revision            | STRUCT of                                               | If the firmware cannot determine the hardware revision value, it shall return zero for both the Major and Minor Revision, else 0 is not a valid value for either the Major or Minor Revision.                                                                                                                                                            |
|              | Major Revision               | USINT                                                   |                                                                                                                                                                                                                                                                                                                                                          |
|              | Minor Revision               | USINT                                                   |                                                                                                                                                                                                                                                                                                                                                          |
| 38           | Reserved                     |                                                         |                                                                                                                                                                                                                                                                                                                                                          |
| 39           | Product Identity Certificate | STRUCT of<br>USINT<br>USINT<br>USINT<br>Padded<br>EPATH | Reserved for security related information                                                                                                                                                                                                                                                                                                                |

**Table 87 – Identity object Vendor ID ranges**

| Range             | Usage                                   |
|-------------------|-----------------------------------------|
| 0                 | Reserved – shall not be used            |
| 0x0001 to 0xFEFF  | Assigned by the ODVA, Inc. organization |
| 0xFF00 to 0xFFFFB | Reserved for future use                 |
| 0xFFFC            | Unspecified IEC 61131-9 vendor          |
| 0xFFFD            | Unspecified Type 9 vendor               |
| 0xFFFE            | Unspecified Type 15 vendor              |
| 0xFFFF            | Reserved – shall not be used            |

**Table 88 – Identity object bit definitions for status instance attribute**

| Bit(s)   | Called                    | Definition                                      |
|----------|---------------------------|-------------------------------------------------|
| 0        | Owned                     | 1 for TRUE, 0 for FALSE                         |
| 1        |                           | Reserved, set to 0                              |
| 2        | Configured                | 1 for TRUE, 0 for FALSE                         |
| 3        |                           | Reserved, set to 0                              |
| 4 to 7   | Extended Device Status    | Vendor specific, or as specified in Table 89    |
| 8        | Minor Recoverable Fault   | 1 for TRUE, 0 for FALSE                         |
| 9        | Minor Unrecoverable Fault | 1 for TRUE, 0 for FALSE                         |
| 10       | Major Recoverable Fault   | 1 for TRUE, 0 for FALSE                         |
| 11       | Major Unrecoverable Fault | 1 for TRUE, 0 for FALSE                         |
| 12 to 15 | Extended Device Status 2  | Reserved (shall be set to 0) or vendor specific |

The values of the status bits 4 to 7 shall be as shown in Table 89.

**Table 89 – Default values for extended device status field (bits 4 to 7) of status instance attribute**

| Value    | Meaning                                                                                                 |
|----------|---------------------------------------------------------------------------------------------------------|
| 0        | Self-Testing or State Unknown                                                                           |
| 1        | Firmware Update in progress                                                                             |
| 2        | At least one faulted I/O connection)                                                                    |
| 3        | No I/O connection established                                                                           |
| 4        | Non-Volatile Configuration Bad                                                                          |
| 5        | Major Fault (either bit 10 or bit 11 in Table 88 is TRUE (1))                                           |
| 6        | At least one I/O connection in run mode                                                                 |
| 7        | At least one I/O connection established, all in idle mode                                               |
| 8        | The Status attribute is not applicable to this instance. Valid only for instances greater than one (1). |
| 9        | reserved, shall not be used                                                                             |
| 10 to 15 | reserved for vendor specific states                                                                     |

**Table 90 – Identity object bit definitions for protection mode instance attribute**

| Bit(s)  | Called                      | Definition                                                                                                                                                                          |
|---------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 to 2  | Implicit Protection Setting | Indicates the current implicit protection setting of the device.<br>Bit 0 = Implicit Protection (clear if Not Protected; set if Protected)<br>Bits 1 to 2 = reserved for future use |
| 3       | Explicit Protection Setting | Indicates the current explicit setting of the device<br>Explicit Protection (clear if Not Protected; set if Protected)                                                              |
| 4 to 15 | Reserved                    | Reserved (shall be set to 0)                                                                                                                                                        |

**Table 91 – Identity object bit definitions for features supported attribute**

| Bit(s)  | Definition                                                                                                                                                                    |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 to 30 | Characteristics, or sets of characteristics that are included in the implementation profile.<br>The exact definition of this field is specific to each implementation profile |
| 31      | Reserved                                                                                                                                                                      |

NOTE The list of public implementation profiles is managed by the ODVA, Inc. organization. Definition of the implementation profiles and their feature values is outside the scope of this document.

#### 4.1.8.2.2 Common services

##### 4.1.8.2.2.1 Get\_Attributes\_All Response

At the **class level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 92.

**Table 92 – 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                      | 1                                  |
| 6            | UINT      | Max ID Number Class Attributes    | 0                                  |
| 7            | UINT      | Max ID Number Instance Attributes | 0                                  |

Default values shall be inserted for all unsupported attributes.

At the **instance level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 93.

**Table 93 – Instance level object/service specific response data of Get\_Attributes\_All**

| Attribute ID | Data type    | Attribute name                  | Default value<br>(if not implemented) |
|--------------|--------------|---------------------------------|---------------------------------------|
| 1            | UINT         | Vendor ID                       |                                       |
| 2            | UINT         | Device Type                     |                                       |
| 3            | UINT         | Product Code                    |                                       |
| 4            | STRUCT of:   | Revision                        |                                       |
|              | USINT        | Major Revision                  |                                       |
|              | USINT        | Minor Revision                  |                                       |
| 5            | WORD         | Status                          |                                       |
| 6            | UDINT        | Serial Number                   |                                       |
| 7            | SHORT STRING | Product Name                    |                                       |
| 8            | USINT        | State                           | 255                                   |
| 9            | UINT         | Configuration Consistency Value | 0                                     |
| 10           | USINT        | Heartbeat Interval              | 0                                     |
| n/a          | STRUCT of:   | Revision 2 indication           |                                       |
|              | USINT        |                                 | 0xFF                                  |
|              | USINT        |                                 | 0xFF                                  |
|              | USINT        |                                 | 0xFF                                  |
| 14           | STRUCT of:   | Semaphore                       |                                       |
|              | UINT         | Client Electronic Key           | 0                                     |
|              | UDINT        |                                 | 0                                     |
|              | ITIME        | Semaphore Timer                 | 0                                     |
| 19           | WORD         | Protection Mode                 | 0                                     |
| 20           | UDINT        | Uptime                          | 0                                     |
| 21           | SHORT_STRING | Catalog Number                  | Zero-length string                    |
| 22           | DATE         | Manufacture Date                | 0                                     |
| 33           | DINT         | Configuration Counter           | 0                                     |
| 34           | STIME        | Configuration Date              | 0                                     |
| 37           | STRUCT of:   | Hardware Revision               |                                       |
|              | USINT        | Major Revision                  | 0                                     |
|              | USINT        | Minor Revision                  | 0                                     |

Default values shall be inserted for all unsupported attributes.

If the Get\_Attributes\_All service returns data after the Heartbeat Interval and the Identity object Revision is 1, the data after the Heartbeat Interval (octet n+4) shall be ignored (since previous editions of the standard contained unparseable content past that point). Attributes 14 and 19 to 22 shall be returned if implemented and Identity object Revision is greater than 1.

Revision 2 Indication shall be used as a marker for Identity Object Revision 2 support. All fields of this structure shall be set to 0xFF. The marker is used to differentiate between devices supporting Revision 2, and Revision 1 devices that could return attribute 11 in the Get\_Attributes\_All response.

#### 4.1.8.2.2.2 Reset service

The Reset common service shall have the object-specific request parameter specified in Table 94.

**Table 94 – Object-specific request parameter for Reset**

| Name       | Data type | Semantics of values                                                                                                                     |
|------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Reset Type | USINT     | Type of Reset. See Table 95.<br>This parameter is optional. If this parameter is omitted, the default value of Power cycle (0) is used. |

**Table 95 – Reset service parameter values**

| Value      | Need in implementation | Name                                                        | Type of Reset                                                                                                                                                                                                                                                                                                                                                              |
|------------|------------------------|-------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0          | Required               | Power cycle                                                 | Emulate as closely as possible cycling power on the item the Identity object represents. This value shall be the default if this parameter is omitted.                                                                                                                                                                                                                     |
| 1          | Optional               | Return to factory defaults                                  | Return as closely as possible to the factory default configuration, then emulate cycling power as closely as possible.                                                                                                                                                                                                                                                     |
| 2          | Optional               | Return to factory defaults except communications parameters | Return as closely as possible to the out-of-box configuration with the exception of communication link parameters and emulate cycling power as closely as possible. The communication link parameters that are to be preserved are defined by each network type.<br><br>See the Reset service of the network specific link object(s) or Table 96 for complete information. |
| 3 to 99    |                        |                                                             | Reserved.                                                                                                                                                                                                                                                                                                                                                                  |
| 100 to 199 |                        |                                                             | Vendor specific                                                                                                                                                                                                                                                                                                                                                            |
| 200 to 255 |                        |                                                             | Reserved                                                                                                                                                                                                                                                                                                                                                                   |

Table 96 indicates the communication link attributes that shall be preserved when the device receives a Reset service with the Reset Type parameter value of 2.

**Table 96 – Communication link attributes that shall be preserved**

| Class                          | Instance number | Attribute               | Attribute ID |
|--------------------------------|-----------------|-------------------------|--------------|
| TCP/IP Interface Object (0xF5) | All except 0    | Configuration Control   | 3            |
|                                |                 | Interface Configuration | 5            |
|                                |                 | Host Name               | 6            |
| Ethernet Link Object (0xF6)    | All except 0    | Interface Control       | 6            |

The Reset common service has no object-specific response parameters.

#### 4.1.8.2.3 Object specific services

##### Flash\_LEDs

The Flash\_LEDs\_RequestPDU body shall be as specified in Table 97.

**Table 97 – Structure of Flash\_LEDs\_RequestPDU body**

| Name | Data type | Semantics of values                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Time | USINT     | <p>The amount of time, in seconds, from when the service was received, that the device flashes its LEDs. If the device is currently flashing, this Time value replaces the time remaining. When the time expires, the device may either complete the current sequence or stop immediately.</p> <p>Values 1-8 are not valid values. The minimum value of 9 allows for a minimum of 6 flashing sequences of the LEDs. Value 0 stops the flashing.</p> |

The Flash\_LEDs\_ResponsePDU body is empty.

### 4.1.8.3 Message Router object specific syntax elements

#### 4.1.8.3.1 Attributes

##### 4.1.8.3.1.1 Class attributes

The Message Router object class attributes shall be as specified in Table 98.

**Table 98 – Message Router object class attributes**

| Attribute ID    | Name                                                         | Data type | Semantics of values                   |
|-----------------|--------------------------------------------------------------|-----------|---------------------------------------|
| 1               | Revision                                                     | UINT      | Revision of this object.<br>Value = 1 |
| 2 – 7, 200, 201 | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                       |

##### 4.1.8.3.1.2 Instance attributes

The format of the Message Router object instance attributes is as specified in Table 99.

**Table 99 – Message Router object instance attributes**

| Attribute ID | Name               | Data type     |
|--------------|--------------------|---------------|
| 1            | Object_List        | STRUCT of     |
|              | Number             | UINT          |
|              | Classes            | ARRAY of UINT |
| 2            | Number available   | UINT          |
| 3            | Number active      | UINT          |
| 4            | Active connections | ARRAY of UINT |

#### 4.1.8.3.2 Common services

##### Get\_Attributes\_All Response

At the **class level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as specified in Table 100.

**Table 100 – Class level object/service specific response data of Get\_Attributes\_All**

| Attribute ID | Data type     | Attribute Name                        | Default Value<br>(if not implemented) |
|--------------|---------------|---------------------------------------|---------------------------------------|
| 1            | UINT          | Revision                              | 1                                     |
| 4            | STRUCT of     | Optional attribute list               |                                       |
|              | UINT          | Number of attributes                  | 0                                     |
|              | ARRAY of UINT | Optional attributes                   | (null)                                |
| 5            | STRUCT of     | Optional service list                 |                                       |
|              | UINT          | Number services                       | 0                                     |
|              | ARRAY of UINT | Optional services                     | (null)                                |
| 6            | UINT          | Maximum ID Number Class Attributes    | 0                                     |
| 7            | UINT          | Maximum ID Number Instance Attributes | 0                                     |

At the **instance level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as specified in Table 101.

**Table 101 – Instance level object/service specific response data of Get\_Attributes\_All**

| Attribute ID | Data type     | Name               | Default Value<br>(if not implemented) |
|--------------|---------------|--------------------|---------------------------------------|
| 1            | STRUCT of     | Object List        |                                       |
|              | UINT          | Number             | 0                                     |
|              | ARRAY of UINT | Classes            | (null)                                |
| 2            | UINT          | Number available   | 0                                     |
| 3            | UINT          | Number active      | 0                                     |
| 4            | ARRAY of UINT | Active connections | (null)                                |

#### 4.1.8.3.3 Object specific services

##### 4.1.8.3.3.1 Symbolic\_Translation

The Symbolic\_Translation\_RequestPDU body shall be as specified in Table 102.

**Table 102 – Structure of Symbolic\_Translation\_RequestPDU body**

| Name             | Data type    |
|------------------|--------------|
| Symbolic Address | Padded EPATH |

The Symbolic\_Translation\_ResponsePDU body shall be as specified in Table 103.

**Table 103 – Structure of Symbolic\_Translation\_ResponsePDU body**

| Name            | Data type    |
|-----------------|--------------|
| Logical Address | Padded EPATH |

The service response may return a status from the list of status codes in Table 104.

**Table 104 – Object specific status for Symbolic\_Translation service**

| General Status Code | Extended Status Code | Error Name                             | Status description                                                                                  |
|---------------------|----------------------|----------------------------------------|-----------------------------------------------------------------------------------------------------|
| 0x20                | 0x00                 | Symbolic Path unknown                  | The Symbolic path is not recognized by the processing node                                          |
| 0x20                | 0x01                 | Symbolic Path destination not assigned | The Symbolic path, although recognized, is not presently associated with a Logical path equivalent. |
| 0x20                | 0x02                 | Symbolic Path segment error            | The symbol identifier or the symbol segment syntax was not understood by the processing node.       |

#### 4.1.8.3.3.2 Send\_Receive\_Fragment

The Send\_Receive\_Fragment\_RequestPDU body for phase 1 shall be as specified in Table 105.

**Table 105 – Structure of Send\_Receive\_Fragment\_RequestPDU body – Phase 1**

| Name                              |                             | Data type      | Semantics of values                                                                                                           |
|-----------------------------------|-----------------------------|----------------|-------------------------------------------------------------------------------------------------------------------------------|
| Fragmentation Request Header      | Fragmentation Version       | USINT          | Currently only version 1 is defined.                                                                                          |
|                                   | Request Fragmentation Flags | SWORD          | See Table 108 and Table 109.                                                                                                  |
|                                   | Reserved                    | UINT           | Shall be set to 0.                                                                                                            |
|                                   | Request Info                | UDINT          |                                                                                                                               |
| Embedded Service Request Fragment |                             | Array of octet | The format of the reassembled Embedded Service Request Fragment parameters is identical to the Message Router request format. |

When a request with the Last flag set is received and the reassembled embedded service request is determined to be valid, the request is sent to the target object, and the session transitions to phase 2.

The Send\_Receive\_Fragment\_RequestPDU body for phase 2 shall be as specified in Table 106.

**Table 106 – Structure of Send\_Receive\_Fragment\_RequestPDU body – Phase 2**

| Name                         |                             | Data type | Semantics of values                                                                                                                  |
|------------------------------|-----------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------|
| Fragmentation Request Header | Fragmentation Version       | USINT     | Currently only version 1 is defined.                                                                                                 |
|                              | Request Fragmentation Flags | SWORD     | See Table 108 and Table 109.<br>For requests to get the next fragment response, this parameter shall always specify Middle fragment. |
|                              | Reserved                    | UINT      | Shall be set to 0.                                                                                                                   |
|                              | Request Info                | UDINT     | For requests to get the next fragment response, this parameter shall be 0.                                                           |

The Send\_Receive\_Fragment\_ResponsePDU body for phase 1 is empty.

The Send\_Receive\_Fragment\_ResponsePDU body for phase 2 shall be as specified in Table 107.

**Table 107 – Structure of Send\_Receive\_Fragment\_ResponsePDU body – Phase 2**

| Name                               |                              | Data type      | Semantics of values                                                                                                          |
|------------------------------------|------------------------------|----------------|------------------------------------------------------------------------------------------------------------------------------|
| Fragmentation Request Header       | Fragmentation Version        | USINT          | Currently only version 1 is defined.                                                                                         |
|                                    | Response Fragmentation Flags | SWORD          | See Table 108 and Table 109.                                                                                                 |
|                                    | Reserved                     | UINT           | Shall be set to 0.                                                                                                           |
|                                    | Response Info                | UDINT          |                                                                                                                              |
| Embedded Service Response Fragment |                              | Array of octet | The format of the combined Embedded Service Response Fragment parameters is identical to the Message Router Response Format. |

If the Message Router in the target detects an error during a fragmentation session, the Send\_Receive\_Fragment\_ResponsePDU body is empty (no Fragmentation Response Header and Embedded Service Response Fragment).

If the target object servicing the embedded request detects an error, the response message returns General Status Code 0x1E (Embedded Service Error). This error response contains response data consisting of the Fragmentation Response Header and Embedded Service Response Fragment. The Embedded Service Response Fragment uses the Message Router response format which includes the General Status Code and Extended Status Code, along with potential response data from the target object.

The Request/Response Fragmentation Flags fields contain flags that control the fragmentation of the request or response as shown in Table 108. The usage of the combined Request/Response Fragmentation Flag values are defined in Table 109.

**Table 108 – Request/Response Fragmentation Flags**

| Bit(s) | Name     | Description                                                                                                                                               |
|--------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0      | First    | Set in the first fragment of a fragmented request or response message.                                                                                    |
| 1      | Last     | Set in the last fragment of a fragmented request or response message.                                                                                     |
| 2      | Abort    | Only valid in request messages, reserved in response messages.<br>Terminate an ongoing fragmentation session. When set, First and Last Flags are ignored. |
| 3-7    | Reserved |                                                                                                                                                           |

**Table 109 – Fragmentation Flags Usage**

| Last | First | Description                                                                                                  |
|------|-------|--------------------------------------------------------------------------------------------------------------|
| 0    | 0     | Middle fragment of a fragmented request or response message.                                                 |
| 0    | 1     | First fragment of a fragmented request or response message.                                                  |
| 1    | 0     | Last fragment of a fragmented request or response message.                                                   |
| 1    | 1     | Single fragment in a fragmented message, i.e. the embedded service request or response fits in one fragment. |

The service response may return a status from the list of status codes in Table 110.

**Table 110 – Object specific status for Send\_Receive\_Fragment service**

| General Status Code | Extended Status Code | Error Name                                                                    | Status description                                                                                                                                                                                                       |                      |                                                  |
|---------------------|----------------------|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|--------------------------------------------------|
| 0x02                | 0x0001               | Request Info Too Large                                                        | The Request Info (total request size) that the client requested is larger than the target can handle.                                                                                                                    |                      |                                                  |
| 0x02                | 0x0002               | Out Of Fragmentation Sessions                                                 | The target is out of available fragmentation sessions.                                                                                                                                                                   |                      |                                                  |
| 0x02                | 0x0003               | Assembled Response Too Large                                                  | The total assembled response is too large for the target to handle.<br><br>For this error, the extended status size is formatted as follows.                                                                             |                      |                                                  |
|                     |                      |                                                                               | Data type                                                                                                                                                                                                                | Value                | Explanation                                      |
|                     |                      |                                                                               | UINT                                                                                                                                                                                                                     | 0x0003               | Extended status code                             |
|                     |                      |                                                                               | UDINT                                                                                                                                                                                                                    | Actual Response Size | Size of assembled response                       |
|                     |                      |                                                                               | UDINT                                                                                                                                                                                                                    | Max Size             | Maximum assembled response the target can handle |
| 0x17                | 0x0001               | No Active Fragmentation Session                                               | A Middle or Last fragment was received before first having started a fragmentation session.                                                                                                                              |                      |                                                  |
| 0x17                | 0x0002               | Middle Fragment Received When Expecting a Last Fragment                       | A Middle fragment was received when expecting a Last fragment.                                                                                                                                                           |                      |                                                  |
| 0x17                | 0x0003               | Last Fragment Specified in Request When Expecting a Middle Fragment – Phase 1 | A fragment with the Last bit set was received when expecting a Middle fragment during phase 1.                                                                                                                           |                      |                                                  |
| 0x17                | 0x0004               | Last Fragment Specified in Request When Expecting Middle Fragment – Phase 2   | A fragment with the Last bit set was received when returning a fragmented response during phase 2.                                                                                                                       |                      |                                                  |
| 0x17                | 0x0005               | Request Info is Out of Sequence                                               | The received offset (Request Info) doesn't align with the expected offset (i.e. the received Request Info isn't equal to the previously received Request Info plus the size of the previously Embedded Service Request). |                      |                                                  |
| 0x17                | 0x0006               | Fragment Received With Invalid Request Info                                   | A fragment with an invalid Request Info parameter set was received. This can be due to either receiving a value of zero when expecting non-zero, or receiving a non-zero value when expecting zero.                      |                      |                                                  |
| 0x17                | 0x0007               | Invalid Size – Too Much Data                                                  | The received offset (Request Info) plus Embedded Service Request Fragment size is greater than the total request size.                                                                                                   |                      |                                                  |
| 0x17                | 0x0008               | Invalid Size – Non-Fragmented Size Mismatch                                   | The received total size in a non-fragmented request (First and Last bits set) does not match the Embedded Service Request Fragment size.                                                                                 |                      |                                                  |
| 0x1E                | N/A                  | Embedded Service Error                                                        | An embedded service resulted in an error                                                                                                                                                                                 |                      |                                                  |
| 0x20                | 0x0001               | Unsupported Version                                                           | The Fragmentation Version parameter contains an unsupported version.                                                                                                                                                     |                      |                                                  |

#### 4.1.8.4 Assembly object specific syntax elements

##### 4.1.8.4.1 Attributes

##### 4.1.8.4.1.1 Class attributes

The format of the Assembly object class attributes shall be as specified in Table 111.

**Table 111 – Assembly object class attributes**

| Attribute ID    | Name                                                         | Data type | Semantics of values                   |
|-----------------|--------------------------------------------------------------|-----------|---------------------------------------|
| 1               | Revision                                                     | UINT      | Revision of this object.<br>Value = 3 |
| 2 – 7, 200, 201 | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                       |

##### 4.1.8.4.1.2 Instance attributes

The format of the Assembly object instance attributes is as specified in Table 112.

**Table 112 – Assembly object instance attributes**

| Attribute ID | Name                      | Data type       | Semantics of values                                                                                                                                                                                                                                                                                   |
|--------------|---------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1            | Number of Members in List | UINT            | Number of members in attribute 2                                                                                                                                                                                                                                                                      |
| 2            | Member List               | ARRAY of STRUCT | This attribute has a complex data type (array of paths)                                                                                                                                                                                                                                               |
|              | Member Data Size          | UINT            | Size of member data or pad in bits                                                                                                                                                                                                                                                                    |
|              | Member Path Size          | UINT            | Size in octets (0 = Empty Path)                                                                                                                                                                                                                                                                       |
|              | Member Path               | Packed EPATH    | Packed path to the member data.<br>See 4.1.9 for definition of Path                                                                                                                                                                                                                                   |
| 3            | Data                      | ARRAY of octet  | Contain all of the member data packed into one array. This data may contain many different data types. For efficiency it is best to keep this data word aligned by packing it on word boundaries and adding padding as needed. This can be accomplished by using "empty paths" (Member Path Size = 0) |
| 4            | Size                      | UINT            |                                                                                                                                                                                                                                                                                                       |
| 5            | Member List Signature     | UDINT           | Used to indicate that the Member List (Attribute 2) has changed. Counter value that increments by one with each change to the Member List attribute and wraps around from 0xFFFFFFFF to 0x00000000. This value shall be retained through power cycles.                                                |

The following rules apply to the Member List attribute.

Members may represent either data elements or pad bits. The members shall be placed into the data block least significant bit first in the order they are listed. The sum of the Member Size value for all members shall be an integer number of octets.

When a member represents a data element, the Member Path field specifies the logical segments that point to the data element. If the Member Size field value is smaller than the defined size of the specified data element, the least significant bits of the corresponding element shall be included. If a Member Size field value is larger than the defined size of the specified data element, the entire data element shall be followed by N pad bits where N is equal to the Member Size minus the size of the data element. Producers shall set pad bits in the Data attribute to 0, consumers shall ignore pad bits.

When a member represents pad bits, the Member Path Size field shall be 0, the Member Path field shall be empty and the number of bits specified by the Member Size field shall specify the number of pad bits to include in the Data attribute. Producers shall set pad bits in the Data attribute to 0, consumers shall ignore pad bits.

#### 4.1.8.4.1.3 Assembly Instance ID ranges

The Assembly Instance ID values shall be as shown in Table 113. Assembly Instance ID = 0x00 is reserved and shall not be used.

**Table 113 – Assembly Instance ID ranges**

| Range (hex)              | Range (decimal)            | Meaning                                           | Quantity      |
|--------------------------|----------------------------|---------------------------------------------------|---------------|
| 0x01 to 0x63             | 1 to 99                    | Open <sup>a</sup>                                 | 99            |
| 0x64 to 0xC7             | 100 to 199                 | Vendor Specific <sup>b</sup>                      | 100           |
| 0xC8 to 0xD1             | 200 to 209                 | Open <sup>a</sup>                                 | 10            |
| 0xD2                     | 210                        | Standard Network Diagnostic assembly <sup>d</sup> | 1             |
| 0xD3 to 0xD7             | 211 to 215                 | Predefined diagnostic assemblies <sup>d</sup>     | 5             |
| 0xD8 to 0xEC             | 216 to 236                 | Reserved <sup>c</sup>                             | 21            |
| 0xED <sup>e</sup>        | 237                        | Open – O2T Heartbeat for Listen Only              | 1             |
| 0xEE <sup>e</sup>        | 238                        | Open – O2T Heartbeat for Input Only               | 1             |
| 0xEF <sup>e</sup>        | 239                        | Open – T2O Heartbeat                              | 1             |
| 0xF0 to 0xFF             | 240 to 255                 | Vendor Specific <sup>b</sup>                      | 16            |
| 0x0100 to 0x02FF         | 256 to 767                 | Open <sup>a</sup>                                 | 512           |
| 0x0300 to 0x4FFF         | 768 to 1 279               | Vendor Specific <sup>b</sup>                      | 512           |
| 0x0500 to 0x7FFF         | 1 280 to 32 767            | Open <sup>a</sup>                                 | 31 488        |
| 0x8000 to 0xFFFF         | 32 768 to 65 535           | Vendor Specific <sup>b</sup>                      | 32 768        |
| 0x00010000 to 0x000FFFFF | 65 536 to 1 048 575        | Open <sup>a</sup>                                 | 983 040       |
| 0x00100000 to 0xFFFFFFFF | 1 048 576 to 4 294 967 295 | Reserved <sup>c</sup>                             | 4 293 918 720 |

<sup>a</sup> Static assemblies defined in device profiles.

<sup>b</sup> Static assemblies and dynamic assemblies

<sup>c</sup> Reserved ranges may be further defined by the ODVA, Inc. organization

<sup>d</sup> Variable assemblies

<sup>e</sup> See 4.1.6.12 for usage definition.

#### 4.1.8.4.1.4 Standard Network Diagnostic assembly – Instance 0xD2 (210)

The object class structures permitted in the Standard Network Diagnostic assembly are specified in Table 114.

**Table 114 – Standard Network Diagnostic assembly content and ordering**

| Diagnostic structure members        | Placement in the assembly | Number of instances required                                                                                                                                   |
|-------------------------------------|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Member List Signature               | 1                         | 1, indicates whether Member List has changed                                                                                                                   |
| Pad                                 | 2                         | 32 bits, shall be zeros                                                                                                                                        |
| Connection Manager Connection Point | 3                         | 1 instance                                                                                                                                                     |
| Ethernet Link Connection Point      | 4                         | 1 instance per Type 2 Ethernet capable port on the device.                                                                                                     |
| TCP/IP Interface Connection Point   | 5                         | 1 instance per Type 2 Ethernet port that has individually configured IP address settings.                                                                      |
| Device Level Ring Connection Point  | 6                         | 1 instance per pair of ports configured for DLR, omitted if no ports are configured for DLR operation.                                                         |
| PRP/HSR Protocol Connection Point   | 7                         | 1 instance per pair of ports configured for PRP, omitted if no ports are configured for PRP.                                                                   |
| Time Sync Connection Point          | 8                         | 1 instance if the device is configured to support Type 2 Time Synchronization, omitted if the device is not configured to support Type 2 Time Synchronization. |
| RSTP Bridge Connection Point        | 9                         | 1 instance per RSTP Bridge, omitted if no RSTP Bridges are configured.                                                                                         |
| RSTP Port Connection Point          | 10                        | 1 instance per RSTP Port, omitted if no RSTP Ports are configured.                                                                                             |

When a device has multiple instances of the same Connection Point, they shall be in instance number order.

NOTE Other content can be added over time to expand this structure.

Position 1 is padded to provide 64-bit alignment. All Connection Point structures are defined to be 64-bit aligned, which ensures that regardless of the quantity of each present (0 to n), the entire assembly structure will maintain 64-bit alignment.

#### 4.1.8.4.2 Common services

##### Create service

The Create common service shall have the object-specific request parameter specified in Table 115.

**Table 115 – Object-specific request parameter for Create**

| Name                       | Data type | Semantics of values                                                                                                                                                                                                                                                                                                  |
|----------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Member List Signature Flag | BOOL      | Indicates if the newly created instance will support the Member List Signature attribute:<br>0 = Omit Member List Signature, default<br>1 = Include Member List Signature, recommended<br><br>This parameter is optional. If this parameter is omitted, the default value of 0 (Omit Member List Signature) is used. |

The Create common service shall have the object-specific response parameter specified in Table 116.

**Table 116 – Object-specific response parameter for Create**

| Name                                                                                                          | Data type                  | Semantics of values                                                                                                                                                                                         |
|---------------------------------------------------------------------------------------------------------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Instance ID                                                                                                   | UINT or UDINT <sup>a</sup> | The Instance ID of the instance that was created<br>The instance ID shall be assigned from one of the vendor specific ranges supported by the device. See Table 113 for the vendor specific ranges allowed. |
| <sup>a</sup> UDINT if dynamic instances can be created with Instance IDs greater than 65 535, UINT otherwise. |                            |                                                                                                                                                                                                             |

**4.1.8.5 Acknowledge Handler object specific syntax elements****4.1.8.5.1 Attributes****4.1.8.5.1.1 Class attributes**

The format of the Acknowledge Handler object class attributes shall be as specified in Table 117.

**Table 117 – Acknowledge Handler object class attributes**

| Attribute ID    | Name                                                         | Data type | Semantics of values                   |
|-----------------|--------------------------------------------------------------|-----------|---------------------------------------|
| 1               | Revision                                                     | UINT      | Revision of this object.<br>Value = 1 |
| 2 – 7, 200, 201 | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                       |

**4.1.8.5.1.2 Instance attributes**

The format of the Acknowledge Handler object instance attributes is as specified in Table 118.

**Table 118 – Acknowledge Handler object instance attributes**

| Attribute ID | Name                              | Data type     | Semantics of values                              |
|--------------|-----------------------------------|---------------|--------------------------------------------------|
| 1            | Acknowledge Timer                 | UINT          | Range 1 to 65 535 ms (0 invalid)<br>Default = 16 |
| 2            | Retry Limit                       | USINT         | Range 0 to 255<br>Default = 1                    |
| 3            | COS Producing Connection Instance | UINT          | Connection Instance ID                           |
| 4            | Ack List Size                     | SWORD         | 0 = Dynamic<br>> 0 = Max number of members       |
| 5            | Ack List                          | STRUCT of     |                                                  |
|              |                                   | SWORD         | Number of members in the array                   |
|              |                                   | ARRAY of UINT | List of Connection Instance IDs                  |
| 6            | Data with Ack Path List Size      | SWORD         | 0 = Dynamic<br>> 0 = Max number of members       |
| 7            | Data with Ack Path List           | STRUCT of     |                                                  |
|              |                                   | SWORD         | Number of members in the array                   |
|              |                                   | ARRAY of      |                                                  |
|              |                                   | UINT          | Connection Instance ID                           |
|              |                                   | USINT         | Path length                                      |
|              |                                   | Padded EPATH  | Path                                             |

#### 4.1.8.5.2 Object specific services

##### 4.1.8.5.2.1 Add\_AckData\_Path

The Add\_AckData\_Path\_RequestPDU body shall be as specified in Table 119.

**Table 119 – Structure of Add\_AckData\_Path\_RequestPDU body**

| Name                   | Data type    |
|------------------------|--------------|
| Connection Instance ID | UINT         |
| Consumer Path Size     | USINT        |
| Consumer Path          | Padded EPATH |

The Add\_AckData\_Path\_ResponsePDU body is empty.

##### 4.1.8.5.2.2 Remove\_AckData\_Path

The Remove\_AckData\_Path\_RequestPDU body shall be as specified in Table 120.

**Table 120 – Structure of Remove\_AckData\_Path\_RequestPDU body**

| Name                   | Data type |
|------------------------|-----------|
| Connection Instance ID | UINT      |

The Remove\_AckData\_Path\_ResponsePDU body is empty.

#### 4.1.8.6 Time Sync object specific syntax elements

##### 4.1.8.6.1 Attributes

##### 4.1.8.6.1.1 Class attributes

The format of the Time Sync object class attributes shall be as specified in Table 121.

**Table 121 – Time Sync object class attributes**

| Attribute ID   | Name                                                         | Data type | Semantics of values                                        |
|----------------|--------------------------------------------------------------|-----------|------------------------------------------------------------|
| 1              | Revision                                                     | UINT      | Revision of this object.<br>Value = 3, 4 or 5 <sup>a</sup> |
| 2, 7, 200, 201 | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                                            |

<sup>a</sup> Revision 4 is required for master capable devices; revision 5 is required for conditional attribute 29 and 31; others may use revision 3, 4 or 5.

##### 4.1.8.6.1.2 Instance attributes

The format of the Time Sync object instance attributes shall be as specified in Table 122.

**Table 122 – Time Sync object instance attributes**

| Attribute ID | Name                            | Data type | Semantics of values                                                                                                                                |
|--------------|---------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 1            | PTPEnable                       | BOOL      | PTP enabled for this device:<br>0 = PTP disabled<br>1 = PTP enabled                                                                                |
| 2            | IsSynchronized                  | BOOL      | Local clock synchronized with master reference clock:<br>0 = local clock not synchronized<br>1 = local clock synchronized                          |
| 3            | SystemTimeMicroseconds          | UTIME     | Current value of system_time in microseconds                                                                                                       |
| 4            | SystemTimeNanoseconds           | STIME     | Current value of system_time in nanoseconds                                                                                                        |
| 5            | OffsetFromMaster                | NTIME     | Offset between local clock and master clock in nanoseconds                                                                                         |
| 6            | MaxOffsetFromMaster             | ULINT     | Maximum offset between local clock and master clock in nanoseconds                                                                                 |
| 7            | MeanPathDelayToMaster           | NTIME     | Mean path delay to master in nanoseconds                                                                                                           |
| 8            | GrandMasterClockInfo            | STRUCT of | Grandmaster Clock information                                                                                                                      |
|              | ClockIdentity                   | USINT[8]  | Unique identifier of the clock. See Table 123 for ClockIdentity encoding                                                                           |
|              | ClockClass                      | UINT      | Class of the clock quality. Table 124 shows the values most likely to be used by the Time sync object. See IEC 61588 for a complete list of values |
|              | TimeAccuracy                    | UINT      | Expected absolute accuracy of the clock relative to the PTP epoch. See Table 125 for TimeAccuracy values                                           |
|              | OffsetScaledLogVariance         | UINT      | Measure of the inherent stability properties of the clock                                                                                          |
|              | CurrentUtcOffset                | UINT      | Current UTC offset in seconds from International Atomic Time (TAI) of the clock. As of 2006-01-01 at 00:00:00 UTC, the offset was 33 seconds       |
|              | TimePropertyFlags               | WORD      | For the value of the TimePropertyFlags see Table 126                                                                                               |
|              | TimeSource                      | UINT      | For the possible TimeSource values see Table 127                                                                                                   |
|              | Priority1                       | UINT      | Relative priority of the grandmaster clock to other clocks in the system. See Priority1 attribute 16                                               |
| 9            | ParentClockInfo                 | STRUCT of | Parent Clock information                                                                                                                           |
|              | ClockIdentity                   | USINT[8]  | Unique identifier of the clock. See Table 123 for ClockIdentity encoding                                                                           |
|              | PortNumber                      | UINT      | PTP Port number                                                                                                                                    |
|              | ObservedOffsetScaledLogVariance | UINT      | Estimated measure of the parent clock's variance as observed by the slave clock                                                                    |
|              | ObservedPhaseChangeRate         | UDINT     | Estimated measure of the parent clock's drift as observed by the slave clock                                                                       |

| Attribute ID | Name                       | Data type          | Semantics of values                                                                                                                                                                                       |
|--------------|----------------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10           | LocalClockInfo             | STRUCT of          | Local Clock information                                                                                                                                                                                   |
|              | ClockIdentity              | USINT[8]           | Unique identifier of the clock See Table 123 for ClockIdentity encoding                                                                                                                                   |
|              | ClockClass                 | UINT               | Class of the clock quality. Table 124 shows the values most likely to be used by the Time sync object. See IEC 61588 for a complete list of values                                                        |
|              | Time Accuracy              | UINT               | Expected absolute accuracy of the clock relative to the PTP epoch. See Table 125 for TimeAccuracy values                                                                                                  |
|              | OffsetScaledLogVariance    | UINT               | Measure of the inherent stability properties of the clock                                                                                                                                                 |
|              | CurrentUtcOffset           | UINT               | Current UTC offset in seconds from International Atomic Time (TAI) of the clock. As of 2006-01-01 at 00:00:00 UTC, the offset was 33 seconds                                                              |
|              | TimePropertyFlags          | WORD               | For the value of the TimePropertyFlags see Table 126                                                                                                                                                      |
|              | TimeSource                 | UINT               | For the possible TimeSource values see Table 127                                                                                                                                                          |
| 11           | NumberOfPorts              | UINT               | Number of PTP ports on the device                                                                                                                                                                         |
| 12           | PortStateInfo              | STRUCT of          | PTP port state information                                                                                                                                                                                |
|              | NumberOfPorts              | UINT               | Number of PTP ports on the device                                                                                                                                                                         |
|              |                            | ARRAY of STRUCT of |                                                                                                                                                                                                           |
|              | PortNumber                 | UINT               | PTP port number                                                                                                                                                                                           |
|              | PortState                  | UINT               | Current state of the PTP port:<br>1 = Initializing<br>2 = Faulty<br>3 = Disabled<br>4 = Listening<br>5 = Pre_Master<br>6 = Master<br>7 = Passive<br>8 = Uncalibrated<br>9 = Slave<br>All other = Reserved |
| 13           | PortEnableCfg              | STRUCT of          | PTP port enable configuration                                                                                                                                                                             |
|              | NumberOfPorts              | UINT               | Number of PTP ports on the device                                                                                                                                                                         |
|              |                            | ARRAY of STRUCT of |                                                                                                                                                                                                           |
|              | PortNumber                 | UINT               | PTP port number                                                                                                                                                                                           |
|              | PortEnable                 | UINT               | PortEnable has the following values (default is "enabled"):<br>1 = PTP port enabled<br>0 = PTP port disabled                                                                                              |
| 14           | PortLogAnnounceIntervalCfg | STRUCT of          | PTP port log announce interval configuration                                                                                                                                                              |
|              | NumberOfPorts              | UINT               | Number of PTP ports on the device                                                                                                                                                                         |
|              |                            | ARRAY of STRUCT of | PTP port log announce interval of each port                                                                                                                                                               |
|              | PortNumber                 | UINT               | PTP port number                                                                                                                                                                                           |
|              | PortLogAnnounceInterval    | UINT               | PTP announce interval between successive "Announce" messages issued by a master clock                                                                                                                     |

| Attribute ID | Name                    | Data type          | Semantics of values                                                                                                                                                                                    |
|--------------|-------------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15           | PortLogSyncIntervalCfg  | STRUCT of          | PTP port log sync interval configuration                                                                                                                                                               |
|              | NumberOfPorts           | UINT               | Number of PTP ports on the device                                                                                                                                                                      |
|              |                         | ARRAY of STRUCT of | PTP port log sync interval of each port                                                                                                                                                                |
|              | PortNumber              | UINT               | PTP port number                                                                                                                                                                                        |
|              | PortLogSyncInterval     | UINT               | PTP sync interval between successive "Sync" messages issued by a master clock                                                                                                                          |
| 16           | Priority1               | USINT              | Best Master ranking of this clock and superseeds the clock quality (class, accuracy, and variance). Values are between 0 and 255, with 0 for the highest priority                                      |
| 17           | Priority2               | USINT              | Best Master ranking of this clock after clock quality (class, accuracy, and variance) has been evaluated and superseeds the tie-breaker. Values are between 0 and 255, with 0 for the highest priority |
| 18           | DomainNumber            | USINT              | PTP clock domain                                                                                                                                                                                       |
| 19           | ClockType               | WORD               | For ClockType values see Table 128                                                                                                                                                                     |
| 20           | ManufactureIdentity     | USINT[4]           | Manufacturer identity of the clock. The first 3 octets specify the IEEE OUI (Organization Unique Id) for the manufacturer. The last octet is reserved                                                  |
| 21           | ProductDescription      | STRUCT of          | Product description of the device that contains the clock, formatted as UTF-8 Unicode with a maximum number of symbols of 64.                                                                          |
|              | Size                    | UDINT              | Size of product description (total number of octets for the Description field) <sup>a</sup>                                                                                                            |
|              | Description             | ARRAY of USINT     | Product description                                                                                                                                                                                    |
| 22           | RevisionData            | STRUCT of          | Revision data of the device that contains the clock, formatted as UTF-8 Unicode with a maximum number of symbols of 32.                                                                                |
|              | Size                    | UDINT              | Size of revision data (total number of octets for the Revision field) <sup>a</sup>                                                                                                                     |
|              | Revision                | ARRAY of USINT     | Revision data                                                                                                                                                                                          |
| 23           | UserDescription         | STRUCT of          | User description of the device that contains the clock, formatted as UTF-8 Unicode with a maximum number of symbols of 128.                                                                            |
|              | Size                    | UDINT              | Size of user description (total number of octets for the Description field) <sup>a</sup>                                                                                                               |
|              | Description             | ARRAY of USINT     | User description                                                                                                                                                                                       |
| 24           | PortProfileIdentityInfo | STRUCT of          | PTP port profile identity information                                                                                                                                                                  |
|              | NumberOfPorts           | UINT               | Number of PTP ports on the device                                                                                                                                                                      |
|              |                         | ARRAY of STRUCT of |                                                                                                                                                                                                        |
|              | PortNumber              | UINT               | PTP port number                                                                                                                                                                                        |
|              | PortProfileIdentity     | USINT[8]           | PTP port profile identity. The profile identifier is contained in the first 6 octets of the array. The last two octets shall be set to zero.                                                           |

| Attribute ID | Name                    | Data type             | Semantics of values                                                                                                                                                                                                                                                             |
|--------------|-------------------------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25           | PortPhysicalAddressInfo | STRUCT of             | PTP port physical address information                                                                                                                                                                                                                                           |
|              | NumberOfPorts           | UINT                  | Number of PTP ports on the device                                                                                                                                                                                                                                               |
|              |                         | ARRAY of<br>STRUCT of |                                                                                                                                                                                                                                                                                 |
|              | PortNumber              | UINT                  | PTP port number                                                                                                                                                                                                                                                                 |
|              | PhysicalProtocol        | USINT[16]             | Physical protocol, expressed in ASCII characters.<br>The maximum number of characters is 16.<br>Unused array elements shall be set to zero.<br>See Table 129 for a list of recommended values.<br><br>EXAMPLE "IEEE 802.3" =<br>49 45 45 45 20 38 30 32 2E 33 00 00 00 00 00 00 |
|              | SizeOfAddress           | UINT                  | Size of the PortPhysicalAddress                                                                                                                                                                                                                                                 |
|              | PortPhysicalAddress     | USINT[16]             | PTP port physical address, expressed as an array of octets.<br>The maximum number of octets is 16.<br>Unused array elements shall be set to zero.<br><br>EXAMPLE MAC address "01 02 03 04 05 06" = 01 02 03 04 05 06 00 00 00 00 00 00 00 00 00 00 (and size is 6 octets)       |
| 26           | PortProtocolAddressInfo | STRUCT of             | PTP port protocol address information                                                                                                                                                                                                                                           |
|              | NumberOfPorts           | UINT                  | Number of PTP ports on the device                                                                                                                                                                                                                                               |
|              |                         | ARRAY of<br>STRUCT of |                                                                                                                                                                                                                                                                                 |
|              | PortNumber              | UINT                  | PTP port number                                                                                                                                                                                                                                                                 |
|              | NetworkProtocol         | UINT                  | NetworkProtocol values<br>1 = UDP/Ipv4 (e.g. Type 2 Ethernet)<br>2 = UDP/Ipv6<br>3 = ISO/IEC/IEEE 8802-3<br>4, 5 = reserved for Type 2 networks<br>0xFFFE = local or unknown protocol<br>all other = reserved                                                                   |
|              | SizeOfAddress           | UINT                  | Size of the PortProtocolAddress                                                                                                                                                                                                                                                 |
|              | PortProtocolAddress     | USINT[16]             | PTP port protocol address (e.g. IP address).<br>The maximum number of octets is 16.<br>Unused array elements shall be set to zero.<br><br>EXAMPLE IP protocol address<br>"192.168.1.2" = C0 A8 01 02 00 00 00 00 00 00 00 00 00 00 00 00 (and size is 4 octets)                 |
| 27           | StepRemoved             | UINT                  | Number of communication paths traversed between the local clock and the grandmaster clock                                                                                                                                                                                       |
| 28           | SytemTimeAndOffset      | STRUCT of             | System time and offset in microseconds                                                                                                                                                                                                                                          |
|              | SystemTime              | UTIME                 | System time in microseconds                                                                                                                                                                                                                                                     |
|              | SystemOffset            | LTIME                 | Offset to the local clock value                                                                                                                                                                                                                                                 |

| Attribute ID                                                                        | Name                             | Data type          | Semantics of values                                                        |
|-------------------------------------------------------------------------------------|----------------------------------|--------------------|----------------------------------------------------------------------------|
| 29                                                                                  | AssociatedInterfaceObjects       | STRUCT of          | Objects associated with PTP ports                                          |
|                                                                                     | NumberOfPorts                    | UINT               | Number of PTP Ports                                                        |
|                                                                                     |                                  | ARRAY of STRUCT of |                                                                            |
|                                                                                     | PortNumber                       | UINT               | PTP port number                                                            |
|                                                                                     | AssociatedObjectPathSize         | USINT              | Number of 16-bit words in the following EPATH                              |
|                                                                                     | AssociatedObject                 | Padded EPATH       | Logical path segments that identify the associated object instance         |
| 30                                                                                  | DefaultLeapSeconds               | UINT               | Default leap seconds                                                       |
| 31                                                                                  | AssociatedInterfaceLabels        | STRUCT of          | Interface labels associated with PTP ports                                 |
|                                                                                     | NumberOfPorts                    | UINT               | Number of PTP Ports                                                        |
|                                                                                     |                                  | ARRAY of STRUCT of |                                                                            |
|                                                                                     | PortNumber                       | UINT               | PTP port number                                                            |
|                                                                                     | AssociatedInterfaceLabelPathSize | USINT              | Number of 16-bit words in the following EPATH                              |
|                                                                                     | AssociatedInterfaceLabel         | Padded EPATH       | Logical path segments that identify the associated object interface labels |
| <sup>a</sup> The number of symbols in the description shall be converted to octets. |                                  |                    |                                                                            |

Table 123 – ClockIdentity encoding for different network implementations

| Network         | Octet 0                                                           | Octet 1 | Octet 2                | Octet 3 | Octet 4                    | Octet 5 | Octet 6 | Octet 7 |
|-----------------|-------------------------------------------------------------------|---------|------------------------|---------|----------------------------|---------|---------|---------|
| Type 2 network  | Encoding is specified by the Type 2 network communication profile |         |                        |         |                            |         |         |         |
| Local or closed | 0xFF                                                              | 0xFF    | Vendor ID <sup>a</sup> |         | Serial Number <sup>a</sup> |         |         |         |
| All others      | See IEC 61588                                                     |         |                        |         |                            |         |         |         |

<sup>a</sup> The Vendor ID and Serial Number are formatted as big endian in the ClockIdentity. For example, a Vendor ID of "0809" and a Serial Number of "03040506" results in a ClockIdentity for a local network of "FF FF 08 09 03 04 05 06".

Table 124 – ClockClass values

| ClockClass                            | Value     |
|---------------------------------------|-----------|
| Primary reference                     | 6         |
| Primary reference (Hold)              | 7         |
| Degraded reference A (Master only)    | 52        |
| Degraded reference B (Master / Slave) | 187       |
| Default                               | 248       |
| Slave only                            | 255       |
| See IEC 61588                         | all other |

**Table 125 – TimeAccuracy values**

| Clock accuracy                    | Value        |
|-----------------------------------|--------------|
| Reserved                          | 0x00 to 0x1F |
| ± 25 ns                           | 0x20         |
| ± 100 ns                          | 0x21         |
| ± 250 ns                          | 0x22         |
| ± 1 µs                            | 0x23         |
| ± 2,5 µs                          | 0x24         |
| ± 10 µs                           | 0x25         |
| ± 25 µs                           | 0x26         |
| ± 100 µs                          | 0x27         |
| ± 250 µs                          | 0x28         |
| ± 1 ms                            | 0x29         |
| ± 2,5 ms                          | 0x2A         |
| ± 10 ms                           | 0x2B         |
| ± 25 ms                           | 0x2C         |
| ± 100 ms                          | 0x2D         |
| ± 250 ms                          | 0x2E         |
| ± 1 s                             | 0x2F         |
| ± 10 s                            | 0x30         |
| > ± 10 s                          | 0x31         |
| Reserved                          | 0x32 to 0x7F |
| For use by alternate PTP profiles | 0x80 to 0xFD |
| Unknown                           | 0xFE         |
| Reserved                          | 0xFF         |

**Table 126 – TimePropertyFlags bit values**

| TimePropertyFlags        | Bit index |
|--------------------------|-----------|
| Leap indicator 61        | 0         |
| Leap indicator 59        | 1         |
| Current UTC offset valid | 2         |
| PTP timescale            | 3         |
| Time traceable           | 4         |
| Frequency traceable      | 5         |

**Table 127 – TimeSource values**

| TimeSource                        | Value        |
|-----------------------------------|--------------|
| ATOMIC CLOCK                      | 0x10         |
| GPS                               | 0x20         |
| TERRESTRIAL RADIO                 | 0x30         |
| PTP                               | 0x40         |
| NTP                               | 0x50         |
| HAND SET                          | 0x60         |
| OTHER                             | 0x90         |
| INTERNAL OSCILLATOR               | 0xA0         |
| For use by alternate PTP profiles | 0xF0 to 0xFE |
| Reserved                          | 0xFF         |

**Table 128 – Types of Clock**

| Bit Index <sup>b</sup>                                                                                                                                                                                                                                                       | Configured clock type        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|
| 0                                                                                                                                                                                                                                                                            | Reserved <sup>a</sup>        |
| 1                                                                                                                                                                                                                                                                            | Reserved <sup>a</sup>        |
| 2                                                                                                                                                                                                                                                                            | Reserved <sup>a</sup>        |
| 3                                                                                                                                                                                                                                                                            | Management node              |
| 4                                                                                                                                                                                                                                                                            | End-to-end transparent clock |
| 5                                                                                                                                                                                                                                                                            | Reserved <sup>a</sup>        |
| 6                                                                                                                                                                                                                                                                            | Boundary clock               |
| 7 <sup>c</sup>                                                                                                                                                                                                                                                               | Ordinary clock               |
| 8 <sup>c</sup>                                                                                                                                                                                                                                                               | Slave only                   |
| 9 to 15                                                                                                                                                                                                                                                                      | Reserved <sup>a</sup>        |
| <sup>a</sup> Reserved bits shall be set to 0.<br><sup>b</sup> A value of 1 for a bit indicates that the capability applies to this node.<br><sup>c</sup> When bit 8 is set, bit 7 shall also be set, as Slave Only functionality is a subset of the Ordinary Clock function. |                              |

**Table 129 – Network protocol to PortPhysicalAddressInfo mapping**

| Network protocol          | PhysicalProtocol                                      | PortPhysicalAddress |
|---------------------------|-------------------------------------------------------|---------------------|
| UDP/IPv4                  | "IEEE 802.3"                                          | Use MAC address     |
| UDP/IPv6                  | "IEEE 802.3"                                          | Use MAC address     |
| IEEE Std 802.3-2018       | "IEEE 802.3"                                          | Use MAC address     |
| Type 2 network            | Specified by the Type 2 network communication profile |                     |
| Local or unknown protocol | Vendor specific                                       | Use vendor specific |

#### 4.1.8.6.1.3 Diagnostic connection points

One diagnostic connection point is defined for the Time Sync object class at the instance level. This connection point is required if the Standard Network Diagnostic Assembly (Instance 0xD2) is implemented (see 4.1.8.4.1.4).

The format of the Time Sync diagnostic connection point is as specified in Table 130.

**Table 130 – Time Sync connection point 1, Standard Network Diagnostics**

| Attribute ID | Name                               | Data type | Attribute size | Size of structure |
|--------------|------------------------------------|-----------|----------------|-------------------|
| 2            | IsSynchronized                     | BOOL      | 1 bit          | 24 octets         |
| n/a          | 63 bits pad, shall be zeros        |           | 63 bits        |                   |
| 5            | OffsetFromMaster                   | NTIME     | 8 octets       |                   |
| 8            | GrandMasterClockInfo.ClockIdentity | SINT[8]   | 8 octets       |                   |

#### 4.1.8.6.2 Common services

##### Get\_Attributes\_All Response

At the **class level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 131.

**Table 131 – Class level object/service specific response data of Get\_Attributes\_All**

| Attribute ID | Data type     | Attribute Name                        | Default Value<br>(if not implemented) |
|--------------|---------------|---------------------------------------|---------------------------------------|
| 1            | UINT          | Revision                              |                                       |
| 2            | UINT          | Max Instance                          | 0                                     |
| 3            | UINT          | Number of Instances                   | 0                                     |
| 4            | STRUCT of     | Optional attribute list               |                                       |
|              | UINT          | Number of attributes                  | 0                                     |
|              | ARRAY of UINT | Optional attributes                   | (null)                                |
| 5            | STRUCT of     | Optional service list                 |                                       |
|              | UINT          | Number services                       | 0                                     |
|              | ARRAY of UINT | Optional services                     | (null)                                |
| 6            | UINT          | Maximum ID Number Class Attributes    | 0                                     |
| 7            | UINT          | Maximum ID Number Instance Attributes | 0                                     |

#### 4.1.8.7 Parameter object specific syntax elements

##### 4.1.8.7.1 Attributes

##### 4.1.8.7.1.1 Class attributes

The format of the Parameter object class attributes shall be as specified in Table 132.

**Table 132 – Parameter object class attributes**

| Attribute ID | Name                                                         | Data type | Semantics of values                                                                                                             |
|--------------|--------------------------------------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------|
| 1            | Revision                                                     | UINT      | Revision of this object.<br>Value = 1                                                                                           |
| 2            | Max Instance                                                 | UINT      | The largest instance number of a created object at this class hierarchy level                                                   |
| 3 – 7        | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                                                                                                                 |
| 8            | Parameter Class Descriptor                                   | WORD      | See Table 133                                                                                                                   |
| 9            | Configuration Assembly Instance                              | UINT      | This attribute shall be set to zero if a configuration assembly is not supported                                                |
| 10           | Native Language                                              | USINT     | 0 = English<br>1 = French<br>2 = Spanish<br>3 = Italian<br>4 = German<br>5 = Japanese<br>6 = Portuguese<br>7 = Mandarin Chinese |
| 200, 201     | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                                                                                                                 |

The Parameter Class Descriptor attribute contains bits to describe parameter characteristics. The bits are supported are specified in Table 133.

**Table 133 – Parameter Class Descriptor bit values**

| Bit | Definition                                                                                                                                                                                                                                     |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0   | Supports Parameter instances<br>1 = Individual Parameter instances ARE supported.<br>0 = NO Parameter instances are supported. Only configuration assembly instance used                                                                       |
| 1   | Supports Full attributes<br>1 = All Full Parameter attributes ARE supported<br>0 = Only Parameter instance stub attributes are supported                                                                                                       |
| 2   | Shall do non-volatile storage save command<br>1 = Shall execute non-volatile storage save command. The Save service of the parameter class is used to save instance parameter values<br>0 = Need not execute non-volatile storage save command |
| 3   | Params are stored in Non-Volatile storage<br>1 = All Full parameters are stored in non-volatile storage<br>0 = Parameters are not stored in non-volatile storage                                                                               |

#### 4.1.8.7.1.2 Instance attributes

The format of the Parameter object instance attributes is as specified in Table 134.

**Table 134 – Parameter object instance attributes**

| Attribute ID | Name                            | Data type                            | Semantics of values                                                                                                                                                                                                              |
|--------------|---------------------------------|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1            | Parameter Value                 | Specified by Data Type (attribute 5) |                                                                                                                                                                                                                                  |
| 2            | Link Path Size                  | USINT                                | Number of octets. If this attribute is 0, then no link is specified.                                                                                                                                                             |
| 3            | Link Path                       | Packed EPATH                         | The Link Path is limited to 255 octets                                                                                                                                                                                           |
| 4            | Descriptor                      | WORD                                 | See Table 135                                                                                                                                                                                                                    |
| 5            | Data Type                       | USINT                                | See 5.2.2                                                                                                                                                                                                                        |
| 6            | Data Size                       | USINT                                |                                                                                                                                                                                                                                  |
| 7            | Parameter Name String           | SHORT_STRING                         | The maximum number of characters is 16                                                                                                                                                                                           |
| 8            | Units String                    | SHORT_STRING                         | The maximum number of characters is 4                                                                                                                                                                                            |
| 9            | Help String                     | SHORT_STRING                         | The maximum number of characters is 64                                                                                                                                                                                           |
| 10           | Minimum Value                   | Specified by Data Type (attribute 5) | See Table 137                                                                                                                                                                                                                    |
| 11           | Maximum Value                   | Specified by Data Type (attribute 5) | See Table 137                                                                                                                                                                                                                    |
| 12           | Default Value                   | Specified by Data Type (attribute 5) | Actual value the parameter should be set to when the user wants the default for the parameter                                                                                                                                    |
| 13           | Scaling Multiplier              | UINT                                 | See Table 138                                                                                                                                                                                                                    |
| 14           | Scaling Divisor                 | UINT                                 | See Table 138                                                                                                                                                                                                                    |
| 15           | Scaling Base                    | UINT                                 | See Table 138                                                                                                                                                                                                                    |
| 16           | Scaling Offset                  | INT                                  | See Table 138                                                                                                                                                                                                                    |
| 17           | Multiplier Link                 | UINT                                 | See Table 139                                                                                                                                                                                                                    |
| 18           | Divisor Link                    | UINT                                 | See Table 139                                                                                                                                                                                                                    |
| 19           | Base Link                       | UINT                                 | See Table 139                                                                                                                                                                                                                    |
| 20           | Offset Link                     | UINT                                 | See Table 139                                                                                                                                                                                                                    |
| 21           | Decimal Precision               | USINT                                | Specifies number of decimal places to use when displaying the scaled engineering value. Also used to determine actual increment value so that incrementing a value causes a change in scaled engineering value to this precision |
| 22           | International Parameter Name    | STRINGI                              | Human readable string representing the parameter name                                                                                                                                                                            |
| 23           | International Engineering Units | STRINGI                              | Engineering units associated with the parameter                                                                                                                                                                                  |
| 24           | International Help String       | STRINGI                              | Human readable string representing the help string                                                                                                                                                                               |

**Table 135 – Semantics of Descriptor Instance attribute**

| Bit | Definition                                       | Meaning                                                                                                                                                                                                                                                                      |
|-----|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0   | Supports settable path                           | Indicates that link path can be set                                                                                                                                                                                                                                          |
| 1   | Supports Get_Enum_String                         | Indicates that enumerated strings can be read with the Get_Enum_String service. If descriptor bit 8 is set, enumerated values outside the min/max range may exist                                                                                                            |
| 2   | Supports scaling                                 | See Table 136                                                                                                                                                                                                                                                                |
| 3   | Supports scaling links                           | See Table 136                                                                                                                                                                                                                                                                |
| 4   | Read only parameter                              | Indicates that the Parameter Value attribute can only be read, and not set                                                                                                                                                                                                   |
| 5   | Monitor parameter                                | Indicates that the Parameter Value attribute is updated in real time by the device                                                                                                                                                                                           |
| 6   | Supports extended precision scaling              | See Table 136                                                                                                                                                                                                                                                                |
| 7   | Supports non-consecutive enumerated strings      | Obsolete                                                                                                                                                                                                                                                                     |
| 8   | Allows both enumeration and min/max range values | Valid for integer data types only. Enumerated values and values in the min/max range are valid. Enumerated values may be inside or outside of the min/max range. If an enumerated value is inside the range, the enumerated string takes precedence over the value           |
| 9   | Non-displayed parameter                          | The configuration tool shall not display this parameter                                                                                                                                                                                                                      |
| 10  | Indirect parameter reference                     | The parameter value represents the instance number of another parameter. The referenced parameter instance shall not be an additional indirect reference. A parameter value of 0 shall indicate no reference. The data type for this parameter shall be USINT, UINT or UDINT |
| 11  | Not addressable                                  | The parameter is not directly addressable from the network by any of the Set_Attribute or Get_Attribute services                                                                                                                                                             |
| 12  | Save supported                                   | This bit, when set, indicates that this parameter can be individually saved by sending the Save service to this parameter instance                                                                                                                                           |
| 13  | Apply supported                                  | This bit, when set, indicates the execution of the Apply service to this parameter instance is required before attribute changes affect instance behavior                                                                                                                    |
| 14  | Write only parameter                             | Indicates that the Parameter Value attribute can only be set, and not read                                                                                                                                                                                                   |
| 15  | Reserved                                         | This shall be set to zero                                                                                                                                                                                                                                                    |

**Table 136 – Descriptor Scaling bits usage**

| Supports extended precision scaling (bit 6) | Supports scaling (bit 2) | Supports scaling links (bit 3) | Meaning                                                                                                                                                                                                             |
|---------------------------------------------|--------------------------|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0                                           | 0                        | n/a                            | No Scaling                                                                                                                                                                                                          |
| 0                                           | 1                        | 0                              | Supports scaling using Formula (1) and Formula (2). The scaling attributes (13-16) values are used.                                                                                                                 |
| 0                                           | 1                        | 1                              | Supports scaling using Formula (1) and Formula (2). For non-zero link attributes (17-20), the indicated parameter value will be used. Otherwise the corresponding scaling attribute (13-16) value is used.          |
| 1                                           | n/a                      | 0                              | Supports extended scaling using Formula (3) and Formula (4). The scaling attribute (13-16) values are used.                                                                                                         |
| 1                                           | n/a                      | 1                              | Supports extended scaling using Formula (3) and Formula (4). For non-zero link attributes (17-20), the indicated parameter value will be used. Otherwise the corresponding scaling attribute (13-16) value is used. |

**Table 137 – Minimum and Maximum Value semantics**

| Data type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Description and semantics                                             | Minimum Value semantics                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Maximum Value semantics                                                                                                         | Required/Optional/Not allowed |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| OCTET                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Bit String – 8 bit length                                             | <p>The least significant bit that is set in the minimum value is the lowest valid bit to be enumerated. The most significant bit that is set in the maximum value is the highest valid bit to be enumerated. A minimum value of zero means that bit 0 is the lowest valid bit to be enumerated. A maximum value of zero is not valid.</p> <p>Example 1 (OCTET type):<br/> Min value = 0b00000001<br/> Max value = 0b00001000<br/> Bits 0 to 3 are enumerated</p> <p>Example 2 (WORD TYPE):<br/> Min value = 0b0000000000000001<br/> Max value = 0b0000000000001011<br/> Bits 0 to 3 are enumerated,<br/> bits 0 and 1 in max value are ignored,<br/> bit 3 in min value is ignored</p> <p>Example 3 (DWORD TYPE):<br/> Min value=<br/> 0b00000000000000000000000000000000<br/> Max value=<br/> 0b000000000000000000000000000001000<br/> Bits 0 to 3 are enumerated,<br/> 0 min value implies bit 0</p> |                                                                                                                                 | Optional                      |
| WORD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Bit String – 16 bit length                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                 |                               |
| DWORD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Bit String – 32 bit length                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                 |                               |
| LWORD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Bit String – 64 bit length                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                 |                               |
| STRING <sup>a</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | String<br>(2 octet length indicator,<br>1 octet per character)        | Minimum string length                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Maximum string length                                                                                                           | Required                      |
| STRING2 <sup>a</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | String<br>(2 octet length indicator,<br>2 octets per character)       | Minimum string length                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Maximum string length                                                                                                           | Required                      |
| STRINGN <sup>a</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | String<br>(2 octet length indicator,<br>N octets per character)       | Minimum string length                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Maximum string length                                                                                                           | Required                      |
| SHORT_STRING <sup>a</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Character string<br>(1 octet length indicator,<br>1 octet characters) | Minimum string length                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Maximum string length                                                                                                           | Required                      |
| STRINGI <sup>a</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | International character string                                        | Minimum length (in characters) of the "stringcontents" component of the string's implicit sequence                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Maximum length (in characters) of the "stringcontents" component of the string's implicit sequence                              | Required                      |
| EPATH <sup>a</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Encoded Path                                                          | Minimum string length                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Maximum string length                                                                                                           | Optional                      |
| ENGUNIT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Engineering Units                                                     | The minimum value for this data type is not defined and shall not be specified in the EDS file or the parameter object instance                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | The maximum value for this data type is not defined and shall not be specified in the EDS file or the parameter object instance | Optional                      |
| All other data types                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                       | The minimum numeric value that may be assigned to the data value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | The maximum numeric value that may be assigned to the data value                                                                | Optional <sup>b</sup>         |
| <p><sup>a</sup> The STRING, STRING2, STRINGN, SHORT_STRING, STRINGI and EPATH data types do not have a minimum or maximum value specification. The minimum and maximum value fields are used to present the minimum and maximum string or path lengths. In these cases, the Data Size parameter is used to represent the number of octets required per character or encoding entry.</p> <p><sup>b</sup> If the Minimum Value and/or Maximum Value is not specified then the minimum and/or maximum value for the parameter data value are as defined in IEC 61158-5-2.</p> |                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                 |                               |

**Table 138 – Scaling Formula attributes**

| Attribute         | Meaning                                                                                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Multiplier Value  | (Mult) Multiplier value for the scaling formula. This value can be based on another parameter                                                                          |
| Divisor Value     | (Div) Divisor value for the scaling formula. This value can be based on another parameter specified in the Divisor Link                                                |
| Base Value        | (Base) Base value for the scaling formula. This value can be based on another parameter specified in the Base Link                                                     |
| Offset Value      | (Offset) Offset value for the scaling formula. This value can be based on another parameter specified in the Offset Link. This attribute is an INT and can be negative |
| Decimal Precision | (Precision) Precision value for the scaling formula. This value cannot be based on another parameter                                                                   |

### Scaling Links

Scaling values (multiplier, divisor, base, and offset) can also be based on other parameters. Scaling Links specify from which instance that scaling value is to be retrieved. If the scaling link attribute is set to 0, then that scaling value is a constant and not linked to another parameter. Table 139 specifies the scaling links.

**Table 139 – Scaling links**

| Attribute       | Meaning                                                                       |
|-----------------|-------------------------------------------------------------------------------|
| Multiplier Link | Specifies the parameter instance from where the Multiplier Value is retrieved |
| Divisor Link    | Specifies the parameter instance from where the Divisor Value is retrieved    |
| Base Link       | Specifies the parameter instance from where the Base Value is retrieved       |
| Offset Link     | Specifies the parameter instance from where the Offset Value is retrieved     |

### Scaling Factor Attributes

The Scaling Factor represents actual UINT and INT parameter values in other formats. Formula (1) shall be used to determine the engineering value from the actual value using scaling.

$$\text{EngValue} = \frac{(\text{ActualValue} + \text{Offset}) \times \text{Mult} \times \text{Base}}{\text{Div}} \quad (1)$$

The engineering value can then be displayed to the user in the terms specified within the Decimal Precision Instance Attribute. The inverse of the Scaling Formula shall be used to determine the actual value from the engineering value using scaling (see Formula (2)).

$$\text{ActualValue} = \frac{(\text{EngValue} \times \text{Div})}{\text{Mult} \times \text{Base}} - \text{Offset} \quad (2)$$

The extended precision scaling factor adds extended precision to the standard scaling factor. Formula (3) shall be used to determine the engineering value from the actual value using extended precision scaling.

$$\text{EngValue} = \frac{(\text{ActualValue} + \text{Offset}) \times \text{Mult} \times \text{Base}}{\text{Div} \times 10^{\text{Precision}}} \quad (3)$$

The engineering value can then be displayed to the user in the terms specified within the Decimal Precision attribute. The inverse of the extended precision scaling formula shall be used to determine the actual value from the engineering value using extended precision scaling (see Formula (4)).

$$\text{ActualValue} = \frac{(\text{EngValue} \times \text{Div} \times 10^{\text{Precision}})}{\text{Mult} \times \text{Base}} - \text{Offset} \quad (4)$$

#### 4.1.8.7.2 Common services

##### **Get\_Attributes\_All Response**

At the **class level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 140. A device is only required to include data in the response data array up to the last implemented attribute.

**Table 140 – Class level object/service specific response data of Get\_Attributes\_All**

| Attribute ID | Data type | Attribute name                  | Default value<br>(if not implemented) |
|--------------|-----------|---------------------------------|---------------------------------------|
| 1            | UINT      | Revision                        | 1                                     |
| 2            | UINT      | Max Instance                    |                                       |
| 8            | WORD      | Parameter Class Descriptor      |                                       |
| 9            | UINT      | Configuration Assembly Instance |                                       |
| 10           | USINT     | Native Language                 | 0                                     |

At the **instance level**, the setting of the Parameter Class Descriptor attribute dictates what attributes are returned by the Get\_Attributes\_All service, as follows:

- if this attribute does not have the "Supports Full Attributes" bit set, only the implemented attributes marked Stub (attribute numbers 1 to 6) are returned by the Get\_Attributes\_All service;
- if this attribute does have the "Supports Full Attributes" bit set, the Get\_Attributes\_All service returns all implemented attributes (numbers 1 to 24).

For Parameter object stubs, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 141.

**Table 141 – Instance level object/service specific response data of Get\_Attributes\_All (Parameter object stub)**

| Attribute ID | Data type                            | Attribute name  | Default value<br>(if not implemented) |
|--------------|--------------------------------------|-----------------|---------------------------------------|
| 1            | Specified by Data Type attribute (5) | Parameter Value |                                       |
| 2            | USINT                                | Link Path Size  |                                       |
| 3            | Packed EPATH                         | Link Path       |                                       |
| 4            | WORD                                 | Descriptor      |                                       |
| 5            | USINT                                | Data Type       |                                       |
| 6            | USINT                                | Data Size       |                                       |

For full Parameter objects, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 142. A device is only required to include data in the response data array up to the last implemented attribute.

**Table 142 – Instance level object/service specific response data of Get\_Attributes\_All (full Parameter object)**

| Attribute ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Data type                            | Attribute name                               | Default value (if not implemented) |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|----------------------------------------------|------------------------------------|
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Specified by Data Type attribute (5) | Parameter Value                              |                                    |
| 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | USINT                                | Link Path Size                               |                                    |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Packed EPATH                         | Link Path                                    |                                    |
| 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | WORD                                 | Descriptor                                   |                                    |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | USINT                                | Data Type                                    |                                    |
| 6                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | USINT                                | Data Size                                    |                                    |
| 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | SHORT_STRING                         | Parameter Name String                        |                                    |
| 8                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | SHORT_STRING                         | Units String                                 |                                    |
| 9                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | SHORT_STRING                         | Help String                                  |                                    |
| 10                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Specified by Data Type attribute (5) | Minimum Value                                |                                    |
| 11                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Specified by Data Type attribute (5) | Maximum Value                                |                                    |
| 12                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Specified by Data Type attribute (5) | Default Value                                |                                    |
| 13                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT                                 | Scaling Multiplier                           |                                    |
| 14                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT                                 | Scaling Divisor                              |                                    |
| 15                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT                                 | Scaling Base                                 |                                    |
| 16                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | INT                                  | Scaling Offset                               |                                    |
| 17                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT                                 | Multiplier Link                              |                                    |
| 18                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT                                 | Divisor Link                                 |                                    |
| 19                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT                                 | Base Link                                    |                                    |
| 20                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | UINT                                 | Offset Link                                  |                                    |
| 21                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | USINT                                | Decimal Precision                            |                                    |
| 22                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | STRINGI                              | International Name Parameter <sup>a</sup>    | 0                                  |
| 23                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | STRINGI                              | International Engineering Units <sup>a</sup> | 0                                  |
| 24                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | STRINGI                              | International Help String <sup>a</sup>       | 0                                  |
| <sup>a</sup> For a device that has implemented the International String attributes, the Parameter Name String character count, the Units String character count and the Help String character count shall be reported as zero. For devices that do not support the International String attributes, each (and every) International String attribute shall be reported as a single octet of zero, or may be missing from the response. If any International String attribute is supported, all three attributes shall be reported in the response. |                                      |                                              |                                    |

#### 4.1.8.7.3 Object specific service

##### Get\_Enum\_String

The Get\_Enum\_String\_RequestPDU body shall be as specified in Table 143.

**Table 143 – Structure of Get\_Enum\_String\_RequestPDU body**

| Name                     | Data type | Semantics of values                                                                                                                                                                                                      |
|--------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enumerated String Number | USINT     | Number of enumerated strings to retrieve.<br><br>Maximum possible range of values = 0 to 255. Value indicates bit offset (relative to zero) for bit enumerations, actual value (relative to zero) for value enumerations |

The Get\_Enum\_String\_ResponsePDU body shall be as specified in Table 144.

**Table 144 – Structure of Get\_Enum\_String\_ResponsePDU body**

| Name              | Data type    | Semantics of values                                          |
|-------------------|--------------|--------------------------------------------------------------|
| Enumerated String | SHORT_STRING | Enumerated strings.<br><br>Maximum number of characters = 16 |

Enumerated strings are human-readable strings that describe either a bit or a value, depending on the parameter's data type. The relationship of the string type with the parameter data type is shown in Table 145.

**Table 145 – Enumerated strings Type versus Parameter data type**

| If the parameter's data type is: | Then the enumerated strings returned are: |
|----------------------------------|-------------------------------------------|
| OCTET, WORD, DWORD, LWORD        | Bit enumerated strings                    |
| USINT, SINT, UINT, INT, BOOL     | Value enumerated strings                  |
| All Other data types             | Invalid for enumeration                   |

If the parameter's data type is OCTET, WORD, DWORD or LWORD, requesting a Get\_Enum\_String service with an enumerated string number of 0 on a parameter returns a SHORT\_STRING that describes bit 0. Requesting a Get\_Enum\_String service with an enumerated string number of 1 on a parameter returns a SHORT\_STRING that describes bit 1. This continues up to the maximum bit number supported by the parameter.

If the parameter's data type is SINT, USINT, INT, UINT or BOOL, requesting a Get\_Enum\_String service with an enumerated string number of 0 on a parameter returns a SHORT\_STRING that describes the value of 0. Requesting a Get\_Enum\_String service with an enumerated string number of 1 returns a SHORT\_STRING that describes the value of 1.

#### **4.1.8.8 Connection Manager object specific syntax elements**

##### **4.1.8.8.1 Attributes**

##### **4.1.8.8.1.1 Class attributes**

The format of the Connection Manager object class attributes shall be as specified in Table 146.

**Table 146 – Connection Manager object class attributes**

| Attribute ID | Name                                                                                                                   | Data type | Semantics of values                   |
|--------------|------------------------------------------------------------------------------------------------------------------------|-----------|---------------------------------------|
| 1            | Revision                                                                                                               | UINT      | Revision of this object.<br>Value = 1 |
| 2 – 7        | These optional Class Attributes are specified in 4.1.10.2.2.                                                           |           |                                       |
| 8, 9         | These attributes are used for safety (see IEC 61784-3-2). They shall not be used if the device is not a safety device. |           |                                       |
| 200, 201     | These optional Class Attributes are specified in 4.1.10.2.2.                                                           |           |                                       |

**4.1.8.8.1.2 Instance attributes**

The format of the Connection Manager object instance attributes is as specified in Table 147.

**Table 147 – Connection Manager object instance attributes**

| Attribute ID | Name                              | Data type     | Semantics of values                                                                                                                                                                        |
|--------------|-----------------------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1            | OpenReqs                          | UINT          |                                                                                                                                                                                            |
| 2            | OpenFormat Rejects                | UINT          |                                                                                                                                                                                            |
| 3            | OpenResource Rejects              | UINT          |                                                                                                                                                                                            |
| 4            | OpenOther Rejects                 | UINT          |                                                                                                                                                                                            |
| 5            | CloseReqs                         | UINT          |                                                                                                                                                                                            |
| 6            | CloseFormat Rejects               | UINT          |                                                                                                                                                                                            |
| 7            | CloseOther Rejects                | UINT          |                                                                                                                                                                                            |
| 8            | ConnTimeouts                      | UINT          | Total number of connection timeouts that have occurred in connections controlled by this Connection Manager.                                                                               |
| 9            | Connection Entry List             | STRUCT of     |                                                                                                                                                                                            |
|              | NumConnEntries                    | UINT          | Number of bits used in the ConnOpenBits element.<br><br>This attribute, divided by 8 and incremented for any remainder, gives the length in octets of the array in the ConnOpenBits field. |
|              | ConnOpenBits                      | ARRAY of BOOL | Bit field. List of connection data. Each bit represents a possible connection.<br><br>0 = No Connection.<br>1 = Connection Established.                                                    |
| 10           | Reserved / Obsolete               |               |                                                                                                                                                                                            |
| 11           | CpuUtilization                    | UINT          | CPU Utilization in tenths of a percent. Range of 0 to 1 000 representing 0,0 % to 100,0 %.                                                                                                 |
| 12           | MaxBuffSize <sup>a</sup>          | UDINT         | Amount of buffer space is in octets.                                                                                                                                                       |
| 13           | BufSize Remaining <sup>a</sup>    | UDINT         | Amount of buffer space is in octets.                                                                                                                                                       |
| 14           | Max Connection Establishment Time | UINT          | Number of milliseconds (1 to 65 535)                                                                                                                                                       |
| 15           | I/O Packets per second            | UDINT         | Current I/O packets per second for I/O traffic                                                                                                                                             |
| 16           | Percent I/O Utilization           | UINT          | Current I/O communications utilization for I/O traffic. Percentage in units of 0,1 %, with 1 100 representing 110,0 % utilization.                                                         |

| Attribute ID                                                                              | Name                        | Data type | Semantics of values                                                                    |
|-------------------------------------------------------------------------------------------|-----------------------------|-----------|----------------------------------------------------------------------------------------|
| 17                                                                                        | Explicit Packets per second | UDINT     | Total number of explicit packets sent and received by this device over the last second |
| 18                                                                                        | Missed I/O Packets          | UDINT     | Running total of missed I/O packets                                                    |
| 19                                                                                        | Type 2 I/O Connections      | UDINT     | Total number of Type 2 I/O connections in use                                          |
| 20                                                                                        | Type 2 Explicit Connections | UDINT     | Total number of Type 2 Explicit Messaging connections in use                           |
| <sup>a</sup> The meaning of, and method of calculating, these values are vendor specific. |                             |           |                                                                                        |

#### 4.1.8.8.1.3 Diagnostic connection points

One diagnostic connection point is defined for the Connection Manager object class at the instance level. This connection point is required if the Standard Network Diagnostic Assembly (Instance 0xD2) is implemented (see 4.1.8.4.1.4).

The format of the Connection Manager diagnostic connection point is as specified in Table 148.

**Table 148 – Connection Manager connection point 1, Standard Network Diagnostics**

| Attribute ID | Name                        | Data type | Attribute size | Size of structure |
|--------------|-----------------------------|-----------|----------------|-------------------|
| 19           | Type 2 I/O Connections      | UDINT     | 4 octets       | 32 octets         |
| 18           | Missed I/O Packets          | UDINT     | 4 octets       |                   |
| 17           | Explicit Packets per second | UDINT     | 4 octets       |                   |
| 15           | I/O Packets per second      | UDINT     | 4 octets       |                   |
| 20           | Type 2 Explicit Connections | UDINT     | 4 octets       |                   |
| 8            | Connection Timeouts         | UINT      | 2 octets       |                   |
| 11           | CPU_Utilization             | UINT      | 2 octets       |                   |
| 16           | Percent I/O Utilization     | UINT      | 2 octets       |                   |
| n/a          | 48 bits pad                 | n/a       | 6 octets       |                   |

#### 4.1.8.8.2 Common services

##### 4.1.8.8.2.1 Get\_Attributes\_All Response

At the **class level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 149.

**Table 149 – 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                          | 1                                  |
| 6            | UINT      | Maximum ID Number Class Attributes    | 0                                  |
| 7            | UINT      | Maximum ID Number Instance Attributes | 0                                  |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in 3.5.3.2.1.

At the **instance level**, the order of the attributes returned in the "Attribute Data" parameter of the Get\_Attributes\_All response shall be as defined in Table 150. This service shall only return values of supported attributes. Therefore, a device can only return values up to the last consecutive supported attribute starting from the first attribute (Attribute 1). Values from any attributes supported beyond these shall be acquired via a Get\_Attribute\_Single service.

**Table 150 – Instance level object/service specific response data of Get\_Attributes\_All**

| Attribute ID | Data type     | Attribute name                    | Default value<br>(if not implemented) <sup>b</sup> |
|--------------|---------------|-----------------------------------|----------------------------------------------------|
| 1            | UINT          | Open Requests                     |                                                    |
| 2            | UINT          | Open Format Rejects               |                                                    |
| 3            | UINT          | Open Resource Rejects             |                                                    |
| 4            | UINT          | Open Other Rejects                |                                                    |
| 5            | UINT          | Close Requests                    |                                                    |
| 6            | UINT          | Close Format Rejects              |                                                    |
| 7            | UINT          | Close Other Rejects               |                                                    |
| 8            | UINT          | Connection Timeouts               |                                                    |
| 9            | STRUCT of:    | Connection Entry List             |                                                    |
|              | UINT          | NumConnEntries <sup>a</sup>       |                                                    |
|              | ARRAY of BOOL | ConnOpenBits                      |                                                    |
| 11           | UINT          | CPU_Utilization                   |                                                    |
| 12           | UDINT         | MaxBuffSize                       |                                                    |
| 13           | UDINT         | BufSize Remaining                 |                                                    |
| 14           | UINT          | Max Connection Establishment Time |                                                    |

<sup>a</sup> This field, divided by 8 and rounded up for any remainder, gives the length of the array (in octets) of the ConnOpenBits field of this structure.

<sup>b</sup> No default values are provided because no unsupported attributes are allowed as described above.

#### 4.1.8.8.2.2 Set\_Attributes\_All Request

At the **instance level**, the order of the attributes returned in the "Attribute Data" parameter of the Set\_Attributes\_All request shall be as defined in Table 151.

**Table 151 – Instance level object/service specific request data of Set\_Attributes\_All**

| Attribute ID | Data type | Attribute name                     | Default values<br>(if not implemented) |
|--------------|-----------|------------------------------------|----------------------------------------|
| 1            | UINT      | Open Requests <sup>a</sup>         | 0                                      |
| 2            | UINT      | Open Format Rejects <sup>a</sup>   | 0                                      |
| 3            | UINT      | Open Resource Rejects <sup>a</sup> | 0                                      |
| 4            | UINT      | Open Other Rejects <sup>a</sup>    | 0                                      |
| 5            | UINT      | Close Requests <sup>a</sup>        | 0                                      |
| 6            | UINT      | Close Format Rejects <sup>a</sup>  | 0                                      |
| 7            | UINT      | Close Other Rejects <sup>a</sup>   | 0                                      |
| 8            | UINT      | Connection Timeouts <sup>a</sup>   | 0                                      |

<sup>a</sup> If this attribute is not supported, the value sent shall be ignored. This condition does not cause the service request to fail.

#### 4.1.8.9 Connection object specific syntax elements

##### 4.1.8.9.1 Attributes

##### 4.1.8.9.1.1 Class attributes

The format of the Connection object class attributes shall be as specified in Table 152.

**Table 152 – Connection object class attributes**

| Attribute ID    | Name                                                         | Data type | Semantics of values                   |
|-----------------|--------------------------------------------------------------|-----------|---------------------------------------|
| 1               | Revision                                                     | UINT      | Revision of this object.<br>Value = 1 |
| 2 – 7, 200, 201 | These optional Class Attributes are specified in 4.1.10.2.2. |           |                                       |

##### 4.1.8.9.1.2 Instance attributes

The format of the Connection object instance attributes is as specified in Table 153.

**Table 153 – Connection object instance attributes**

| Attr ID | Name                            | Data type    | Semantics of values                                                                                                                 |
|---------|---------------------------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------|
| 1       | State                           | USINT        | State of the object.                                                                                                                |
| 2       | Instance_type                   | USINT        | Indicates either I/O or Messaging Connection.                                                                                       |
| 3       | TransportClass_trigger          | SWORD        | Defines behavior of the Connection.                                                                                                 |
| 4       | DN_produced_connection_id       | UINT         | Placed in ISO 11898-1 CAN Identifier Field when the Connection transmits on a CAN-based subnet.                                     |
| 5       | DN_consumed_connection_id       | UINT         | ISO 11898-1 CAN Identifier Field value that denotes message to be received on a CAN-based subnet.                                   |
| 6       | DN_initial_comm_characteristics | SWORD        | Defines the Message Group(s) across which productions and consumptions associated with this Connection occur on a CAN-based subnet. |
| 7       | Produced_connection_size        | UINT         | Maximum number of octets transmitted across this Connection.                                                                        |
| 8       | Consumed_connection_size        | UINT         | Maximum number of octets received across this Connection.                                                                           |
| 9       | Expected_packet_rate            | UINT         | Defines timing associated with this Connection.                                                                                     |
| 10      | MSG_produced_connection_id      | UDINT        | Identifies the message sent on the subnet by this connection.                                                                       |
| 11      | MSG_consumed_connection_id      | UDINT        | Identifies the message received from the subnet for this connection.                                                                |
| 12      | Watchdog_timeout_action         | USINT        | Defines how to handle Inactivity/Watchdog timeouts.                                                                                 |
| 13      | Produced_connection_path_length | UINT         | Number of octets in the produced_connection_path attribute.                                                                         |
| 14      | Produced_connection_path        | Packed EPATH | Specifies the application object(s) whose data is to be produced by this Connection object. See 4.1.9.                              |
| 15      | Consumed_connection_path_length | UINT         | Number of octets in the consumed_connection_path attribute.                                                                         |
| 16      | Consumed_connection_path        | Packed EPATH | Specifies the application object(s) that are to receive the data consumed by this Connection object. See 4.1.9.                     |
| 17      | Production_inhibit_time         | UINT         | Defines minimum time between new data production.                                                                                   |

| Attr ID | Name                          | Data type                          | Semantics of values                                                                                                       |
|---------|-------------------------------|------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| 18      | Connection_timeout_multiplier | USINT                              | Specifies the multiplier applied to the expected_packet_rate value to derive the value for the Inactivity/Watchdog Timer. |
| 19      | Connection_binding_list       | STRUCT<br>UINT<br>Array of<br>UINT | List of I/O connection instances bound to this instance.                                                                  |

Detailed descriptions of the Connection object instance attributes are provided below. The defined default attribute values shall be used when no other internal and/or system-defined rules exist.

### State

This attribute defines the current state of the Connection instance. Table 154 defines the possible states and assigns a value used to indicate that state.

**Table 154 – Values assigned to the state attribute**

| Value                                                                                    | State Name                             | Description                                                                                                                                                                                                                                                                                    |
|------------------------------------------------------------------------------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00                                                                                       | Non-existent                           | The Connection has yet to be instantiated.                                                                                                                                                                                                                                                     |
| 01                                                                                       | Configuring                            | The Connection has been instantiated and is waiting for the following events to occur: (1) to be properly configured and (2) to be told to apply the configuration.                                                                                                                            |
| 02                                                                                       | Waiting For Connection ID <sup>a</sup> | See definition in Type 2 network communication profile.                                                                                                                                                                                                                                        |
| 03                                                                                       | Established                            | The Connection has been validly/fully configured and the configuration has been successfully applied.                                                                                                                                                                                          |
| 04                                                                                       | Timed Out                              | If a Connection object experiences an Inactivity/Watchdog timeout, then a transition may be made to this state. See the watchdog_timeout_action attribute description and the description of the Inactivity/Watchdog Timer in IEC 61158-5-2, 6.2.3.2.1.7 (Connection Timing) for more details. |
| 05                                                                                       | Deferred Delete <sup>a</sup>           | See definition in Type 2 network communication profile.                                                                                                                                                                                                                                        |
| 06                                                                                       | Closing                                | A Bridged Connection object has received, and is processing, a Forward Close from the Connection Manager. The deletion of the connection does not occur until after a successful Forward Close response has been received from the target node.                                                |
| <sup>a</sup> This value is only used by a specific Type 2 network communication profile. |                                        |                                                                                                                                                                                                                                                                                                |

**Important:** A dynamically created connection instance is the child of the Explicit Messaging connection across which it was created. Different subnet types may provide other mechanisms for creating a connection that imply a particular parent-child relationship. See network-specific specifications for details.

**Important:** All resources associated with a Connection Instance (A) that has been dynamically created across an Explicit Messaging Connection (B) shall be released if the Explicit Messaging Connection (B) times out prior to the dynamically created Connection Instance (A) transitioning to the Established state.

**Important:** When a transition is made to the Established state, all timers associated with the Connection object are activated (see Connection Timing in IEC 61158-5-2, 6.2.3.2.1.7)

### **Instance\_type**

This attribute defines the instance type (see Table 155).

**Table 155 – Values assigned to the instance\_type attribute**

| Value | Meaning                                                                                                                                                                                                                                                                                                                                                               |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00    | Explicit Messaging. This Connection Instance represents one of the end-points of an Explicit Messaging Connection. An Explicit Messaging Connection is dynamically created by sending the <i>Open Explicit Messaging Connection Request</i> to the Connection Class.                                                                                                  |
| 01    | I/O. This Connection Instance represents one of the end-points of an I/O Connection. An I/O Connection is dynamically created by sending a <i>Create Request</i> to the Connection Class.                                                                                                                                                                             |
| 02    | Bridged. This Connection Instance represents an intermediate 'hop' of a bridged I/O or Explicit Messaging connection. A pair of Bridged Connection objects (one for each subnet bridged between) are dynamically created by a node after successfully receiving a Forward Open service to the Connection Manager object when this node is not the end point (target). |

### **TransportClass\_trigger**

Defines whether this is a producing only, consuming only, or both producing and consuming connection. If this end point is to perform a data production, this attribute also defines the event that triggers the production. The eight (8) bits are divided as shown in Figure 7.

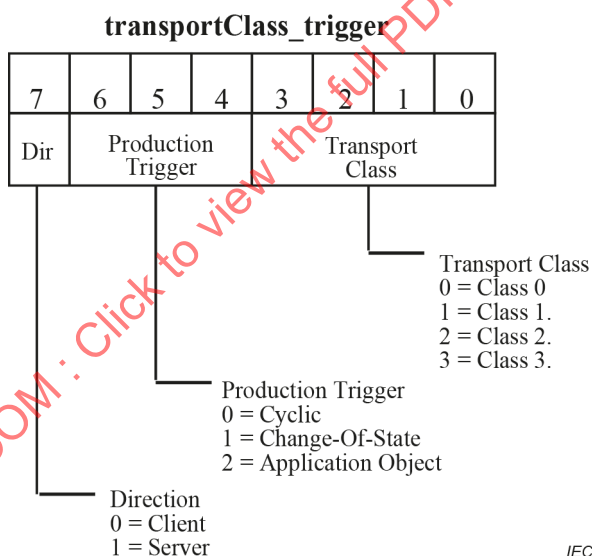


**Figure 7 – Transport Class Trigger attribute**

The **Direction bit** of the transportClass\_trigger octet indicates whether the end-point is to act as the Client or the Server on this connection (see Table 156).

**Table 156 – Possible values within Direction Bit**

| Value | Meaning |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0     | Client  | This end-point provides the Client behavior associated with this Connection. Additionally, this value indicates that the <i>Production Trigger</i> bits within the <i>transportClass_trigger</i> octet contain the description of when the Client is to produce the message associated with this connection. Client connections with production trigger value of 0 or 1 (Cyclic or Change-of-State) shall produce immediately after transitioning to the Established state.                                                           |
| 1     | Server  | This end-point provides the Server behavior associated with this Connection. In addition, this value indicates that the <i>Production Trigger</i> bits within the <i>transportClass_trigger</i> octet are to be IGNORED. The <i>Production Trigger</i> bits are ignored due to the fact that a Server end-point <i>reacts</i> to the transmission from the Client. The only means by which a Server end-point is triggered to transmit is when this <i>reaction</i> calls for the production of a message (Transport Classes 2 or 3). |

Table 157 lists the values that are possible within the **Production Trigger** bits of the **transportClass\_trigger** attribute.

**Table 157 – Possible values within Production Trigger Bits**

| If the value is: | Then the Production of a message is: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0                | Cyclic                               | The expiration of the Transmission Trigger Timer triggers the data production. See IEC 61158-5-2, 6.2.3.2.1.7 Connection Timing for a detailed description of the Transmission Trigger Timer.                                                                                                                                                                                                                                                                                                                  |
| 1                | Change-Of-State                      | Production occurs when a change-of-state is detected by the application object. The term used to describe this production is "new data".<br><br>NOTE The consuming end-point could have been configured to expect the packet at a certain rate, regardless of the triggering mechanism at the producing end-point.<br><br>See the description of the <i>expected_packet_rate</i> attribute of a Connection object and the description of Connection Timing in IEC 61158-5-2, 6.2.3.2.1.7 for more information. |
| 2                | Application Object Triggered         | The application object decides when to trigger the production. The term used to describe this production is "new data".<br><br>NOTE The consuming end-point could have been configured to expect the packet at a certain rate, regardless of the triggering mechanism at the producing end-point.<br><br>See the description of the <i>expected_packet_rate</i> attribute of a Connection object and the description of Connection Timing in IEC 61158-5-2, 6.2.3.2.1.7 for more information.                  |
| 3 - 7            | Reserved                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

Table 158 lists possible values within the **Transport Class** nibble of the **transportClass\_trigger** attribute. Behaviors resulting from these particular values are illustrated in the series of figures that follow the table.

**Table 158 – Possible values within Transport Class Bits**

| Value  | Meaning              |                                                                                                                                                                                                                                                                                                                                                                            |
|--------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0      | Transport Class 0    | Based on the value within the <i>Dir</i> bit, this connection end-point will be a producing only OR consuming only end-point. Upon application of this Connection instance, the module instantiates either a Link Producer ( <i>Dir</i> bit = Client, producing only) or a Link Consumer ( <i>Dir</i> bit = Server, consuming only) to be associated with this Connection. |
| 1      | Transport Class 1    |                                                                                                                                                                                                                                                                                                                                                                            |
| 2      | Transport Class 2    | Indicates that the module will both produce AND consume across this connection. The Client end-point generates the first data production that is consumed by the Server, which causes the Server to return a production that is consumed by the Client.                                                                                                                    |
| 3      | Transport Class 3    |                                                                                                                                                                                                                                                                                                                                                                            |
| 4 – 6  | Reserved (obsoleted) |                                                                                                                                                                                                                                                                                                                                                                            |
| 7 – 15 | Reserved             |                                                                                                                                                                                                                                                                                                                                                                            |

A 16-bit sequence count value is prepended to all Class 1, 2, and 3 transports. This value is used to detect delivery of duplicate data packets. Sequence count values are initialized on the first message production and determined by the transport and trigger type on subsequent message productions. A resend of old data to maintain the connection shall not cause the sequence count to change and a consumer shall ignore data when it is received with a duplicate sequence count. Consuming applications can use this mechanism to distinguish between new samples and old samples that were sent to maintain the connection.

For Class 2 and 3 transports, when a request is received with a duplicate sequence count the receiving device shall respond with the original response. This mechanism can be used to keep Class 2 and 3 transports alive and to deal with cases of dropped packets.

Table 159, Table 160 and Table 161 summarize the valid combinations of **Production Trigger** and **Transport Class** and the associated Client and Server behaviors. See IEC 61158-5-2, 6.2.3.2.1.7 for a description of the **Transmission Trigger Timer**.

Table 159 summarizes the valid values for the **transportClass\_trigger** attribute of a Connection Instance:

**Table 159 – TransportClass\_Trigger attribute values summary**

| TransportClass_trigger bits | Meaning                                                                                                                                                                              |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 xxx 0000                  | Direction = Server, Production Trigger = IGNORED, Transport Class = 0.                                                                                                               |
| 1 xxx 0001                  | Direction = Server, Production Trigger = IGNORED, Transport Class = 1.                                                                                                               |
| 1 xxx 0010                  | Direction = Server, Production Trigger = IGNORED, Transport Class = 2.                                                                                                               |
| 1 xxx 0011                  | Direction = Server, Production Trigger = IGNORED, Transport Class = 3. This is the value assigned to this attribute within the Server end-point of an Explicit Messaging Connection. |
| 0 000 0000                  | Direction = Client, Production Trigger = Cyclic, Transport Class = 0.                                                                                                                |
| 0 000 0001                  | Direction = Client, Production Trigger = Cyclic, Transport Class = 1.                                                                                                                |
| 0 000 0010                  | Direction = Client, Production Trigger = Cyclic, Transport Class = 2.                                                                                                                |
| 0 000 0011                  | Direction = Client, Production Trigger = Cyclic, Transport Class = 3.                                                                                                                |
| 0 001 0000                  | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 0.                                                                                                       |
| 0 001 0001                  | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 1.                                                                                                       |
| 0 001 0010                  | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 2.                                                                                                       |
| 0 001 0011                  | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 3.                                                                                                       |
| 0 010 0000                  | Direction = Client, Production Trigger = Application object, Transport Class = 0.                                                                                                    |
| 0 010 00001                 | Direction = Client, Production Trigger = Application object, Transport Class = 1                                                                                                     |

| TransportClass_trigger bits | Meaning                                                                                                                                                                                         |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 010 0010                  | Direction = Client, Production Trigger = Application object, Transport Class = 2.                                                                                                               |
| 0 010 0011                  | Direction = Client, Production Trigger = Application object, Transport Class = 3. This is the value assigned to this attribute within the Client end-point of an Explicit Messaging Connection. |
| 1 111 1111                  | Default value assigned to this attribute within an I/O Connection.                                                                                                                              |

Table 160 summarizes client behavior for Class 0 in regards to message production for the different trigger types.

**Table 160 – Transport Class 0 client behavior summary**

| Trigger type    | Production triggered by    | Repeated production to maintain connection |
|-----------------|----------------------------|--------------------------------------------|
| Cyclic          | Transmission Timer (API)   | n/a                                        |
| Change of State | Detected data change       | Transmission Timer (API)                   |
| Application     | Application update of data | Transmission Timer (API)                   |

Table 161 summarizes client behavior for Classes 1, 2 and 3 in regards to message production and sequence count for the different trigger types.

**Table 161 – Transport Class 1, 2 and 3 client behavior summary**

| Trigger type    | Sequence count incremented by | Production triggered by    | Repeated production to maintain connection |
|-----------------|-------------------------------|----------------------------|--------------------------------------------|
| Cyclic          | Application update of data    | Transmission Timer (API)   | n/a                                        |
| Change of State | Detected data change          | Detected data change       | Transmission Timer (API)                   |
| Application     | Application update of data    | Application update of data | Transmission Timer (API)                   |

### ***DN\_produced\_connection\_id***

This attribute shall not be used, unless otherwise specified in the network communication profile of the subnet.

It contains the Connection ID to be associated with transmissions sent across this connection (if any), i.e. the value that will be specified in the CAN Identifier Field of ISO 11898-1 when this Connection transmits. See IEC 62026-3:2014, 5.1.2 for a description of how the CAN Identifier Field is used. This value is loaded directly into the associated Link Producer's connection\_id attribute. Values are defined in Table 162.

**Table 162 – Values defined for the DN\_produced\_connection\_id attribute**

| Value            | Meaning                                                                                                                                                                         |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x0000 - 0x07F0  | The value to be placed in the CAN Identifier Field when this Connection transmits.                                                                                              |
| 0x0800 - 0xFFFFE | Reserved                                                                                                                                                                        |
| 0xFFFF           | Default value assigned to this attribute within an I/O Connection. This attribute will retain this value if this Connection instance is not producing any data (consumer only). |

### ***DN\_consumed\_connection\_id***

This attribute shall not be used, unless otherwise specified in the network communication profile of the subnet.

It contains the Connection ID which identifies messages to be received across this connection (if any), i.e. the CAN Identifier Field value that is associated with messages this Connection object receives. See IEC 62026-3:2014, 5.1.2 for a description of how the CAN Identifier Field is used. This value is loaded directly into the associated Link Consumer's connection\_id attribute. Values are defined in Table 163.

**Table 163 – Values defined for the DN\_consumed\_connection\_id attribute**

| Value           | Meaning                                                                                                                                                                         |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x0000 - 0x07F0 | The value that identifies messages to be consumed. This will be specified in the CAN Identifier Field of messages that are to be consumed.                                      |
| 0x0800 - 0xFFFE | Reserved                                                                                                                                                                        |
| 0xFFFF          | Default value assigned to this attribute within an I/O Connection. This attribute will retain this value if this Connection instance is not consuming any data (producer only). |

### ***DN\_initial\_comm\_characteristics***

This attribute shall not be used, unless otherwise specified in the network communication profile of the subnet.

It defines the Message Group(s) across which productions and consumptions associated with this Connection occur.

This octet is divided into two nibbles, as show in Figure 8.

|                                    |   |   |   |                                     |   |   |   |
|------------------------------------|---|---|---|-------------------------------------|---|---|---|
| 7                                  | 6 | 5 | 4 | 3                                   | 2 | 1 | 0 |
| Initial Production Characteristics |   |   |   | Initial Consumption Characteristics |   |   |   |

**Figure 8 – DN\_initial\_comm\_characteristics attribute format**

Table 164 lists the values that are possible within the Initial Production Characteristics nibble (upper nibble) of the DN\_initial\_comm\_characteristics attribute.

**Table 164 – Values for the Initial Production Characteristics nibble**

| Value     | Meaning                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x0       | Produce across Message Group 1               | The production associated with this Connection is to take place across Message Group 1. The producing module generates the Connection ID value and loads it into the Connection object's DN_produced_connection_id attribute. The producing module allocates a Message ID from its Group 1 Message ID pool and combines this with its Source MAC ID to generate the Connection ID. The numerically lowest available Group 1 Message ID is to be used in generating the DN_produced_connection_id attribute value. This value shall also be loaded into the corresponding DN_consumed_connection_id attribute(s) associated with the consuming Connection object(s). |
| 0x1       | Produce across Message Group 2 (Destination) | The production associated with this Connection is to take place across Message Group 2. Additionally, the intended recipient's MAC ID (Destination MAC ID) is to be placed within the MAC ID component of the Group 2 Identifier Field. In this case, the consuming module generates the Connection ID value to be associated with transmissions across this connection. When the consuming module has generated this value and loaded it into the appropriate Connection object's DN_consumed_connection_id attribute, it can be read and subsequently loaded into the producing Connection object's DN_produced_connection_id attribute.                          |
| 0x2       | Produce across Message Group 2 (Source)      | The production associated with this Connection is to take place across Message Group 2. In addition, the producing module's MAC ID (Source MAC ID) is to be placed within the MAC ID component of the Group 2 Identifier. In this case, the producing module generates the Connection ID value and loads it into the Connection object's DN_produced_connection_id attribute. The numerically lowest available Group 2 Message ID is to be used in generating the DN_produced_connection_id attribute value. This value shall also be loaded into the corresponding DN_consumed_connection_id attribute(s) associated with the consuming Connection object(s).      |
| 0x3       | Produce across Message Group 3               | The production associated with this Connection is to take place across Message Group 3. The producing module generates the Connection ID value and loads it into the Connection object's DN_produced_connection_id attribute. The producing module allocates a Message ID from its Group 3 Message ID pool and combines this with its Source MAC ID to generate the Connection ID. The numerically lowest available Group 3 Message ID is to be used in generating the DN_produced_connection_id attribute value. This value shall also be loaded into the corresponding DN_consumed_connection_id attribute(s) associated with the consuming Connection object(s). |
| 0x4 - 0xE | Reserved                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0xF       | Default value                                | The default value assigned to the Initial Production Characteristics nibble within an I/O Connection.<br><br>NOTE If this is a consuming only I/O Connection, then the default value remains in this nibble. Explicit Messaging Connection objects automatically configure this attribute when the Connection is established.                                                                                                                                                                                                                                                                                                                                       |

Table 165 lists the possible values within the Initial Consumption Characteristics nibble (lower nibble) of the DN\_initial\_comm\_characteristics attribute.

**Table 165 – Values for the Initial Consumption Characteristics nibble**

| Value     | Meaning                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x0       | Consume a Group 1 Message               | The message to be consumed will be transmitted across Message Group 1. The producing module generates the Connection ID value. This value shall be loaded into the DN_consumed_connection_id attribute associated with the consuming Connection object(s).                                                                                                                                                                                                                                                                                                             |
| 0x1       | Consume a Group 2 Message (Destination) | The message to be consumed will be transmitted across Message Group 2. The intended recipient's MAC ID (Destination MAC ID) is specified within the Group 2 Identifier. The consuming module generates the Connection ID value and loads it into the DN_consumed_connection_id attribute associated with this Connection object. The numerically lowest available Group 2 Message ID is to be used in generating the DN_consumed_connection_id attribute value. This value shall be loaded into the producing Connection object's DN_produced_connection_id attribute. |
| 0x2       | Consume a Group 2 Message (Source)      | The message to be consumed will be transmitted across Message Group 2. The transmitting module's MAC ID (Source MAC ID) is specified within the Group 2 Identifier. In this case, the producing module generates the Connection ID value and loads it into the Connection object's the DN_produced_connection_id attribute. This value shall be loaded into the DN_consumed_connection_id attribute associated with the consuming Connection object(s).                                                                                                                |
| 0x3       | Consume a Group 3 Message               | The message to be consumed will be transmitted as a Group 3 Message. The producing module generates the Connection ID value. The Connection ID value shall be loaded into this Connection object's DN_consumed_connection_id attribute.                                                                                                                                                                                                                                                                                                                                |
| 0x4 - 0xE | Reserved                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 0xF       | Default value                           | The default value assigned to the Initial Consumption Characteristics nibble within an I/O Connection.<br><br><b>NOTE</b> If this is a producing only I/O Connection, then the default value remains in this nibble. Explicit Messaging Connections automatically configure this attribute when the Connection is established.                                                                                                                                                                                                                                         |

**Important:** The module that generates a Connection ID shall guarantee that it does not allocate the Message ID/MAC ID pair in such a way that two separate modules are capable of transmitting identical bit patterns within the Identifier Field.

#### ***Produced\_connection\_size***

The meaning of this attribute is different for Explicit Messaging Connections than it is for I/O Connections. If the subnet defines a fragmentation protocol and the device supports fragmentation, this size may be larger than the largest frame size.

#### ***For Explicit Messaging Connections:***

This attribute signifies the maximum number of Message Router Request/Response Data octets (see 4.1.8) that a device is able to transmit across this Connection. Devices that place a known limit on the maximum amount of Message Router Request/Response Data that can be transmitted in a single message, or single fragmented series initialize this attribute accordingly. Devices that cannot or do not predefine an up-front transmit limit place the value 0xffff into this attribute (there may still be a limit, however, it is not known in advance).

**Important:** Due to the nature of Explicit Messaging, the length of Explicit Messages will fluctuate over the lifetime of a connection. Explicit Messaging Connections perform fragmentation based on the length of the current message to transmit where the subnet type may define a fragmentation protocol.

***For I/O Connections:***

If the transportClass\_trigger indicates that this Connection instance is to produce, then this attribute defines the maximum amount of I/O data that may be produced as a single unit across this connection. The amount of I/O to be transmitted at any given point in time can be less than or equal to the connection\_size attribute.

This attribute defaults to zero (0) within an I/O Connection. If the subnet type supports fragmentation and this attribute is set to a value greater than the largest payload in an I/O Connection, then the Connection will break up the data into multiple fragments.

**Important:** Fragmentation within I/O Connections is performed based on the value within this attribute, regardless of the current amount of data to transmit.

**Important:** I/O Messages that contain no Application I/O Data and were configured to contain data (via produced\_connection\_size being greater than zero (0)) are defined to indicate a No Data event for the receiving application object(s). The behavior of an application object upon detection of the No Data event is application object specific.

***Consumed\_connection\_size***

The meaning of this attribute is different for Explicit Messaging Connections than it is for I/O Connections. If the subnet defines a fragmentation protocol and the device supports fragmentation, this size may be larger than the largest frame size. See the network specific adaptation specifications for more details.

***For Explicit Messaging Connections:***

This attribute signifies the maximum number of Message Router Request/Response Data octets that a device is able to receive across this Connection.

Devices that place a known limit on the maximum amount of Message Router Request/Response Data that can be received in a single message, or single fragmented series initialize this attribute accordingly. Devices that cannot or do not predefine an up-front receive limit place the value 0xffff into this attribute (there may still be a limit, however, it is not known in advance). Because of the nature of Explicit Messaging, the length of Explicit Messages will fluctuate over the lifetime of a connection.

***For I/O Connections:***

If the transportClass\_trigger attribute indicates that this Connection is to consume, then this attribute defines the maximum amount of data that may be received as a single unit across this connection. The actual amount of I/O data received at any given time can be less than or equal to the connection\_size attribute.

This attribute defaults to zero (0) within an I/O Connection. If the subnet type supports fragmentation and this attribute is set to a value greater than the largest payload in an I/O Connection, then the Connection will process the fragmentation protocol. See the network specific adaptation specifications for more details.

The length of an I/O Message shall be less than or equal to this attribute for an I/O Connection object to receive it as a valid message. If an I/O Connection object receives a message whose length is greater than this attribute, then it immediately discards the message and discontinues any subsequent processing.

NOTE With respect to the Server end-point of a Transport Class 2 or 3 Connection, this error condition (too much data) results in no response being transmitted and the watchdog timer is not kicked.

***Expected\_packet\_rate***

This attribute is used to generate the values loaded into the **Transmission Trigger Timer** and the **Inactivity/Watchdog Timer**. See IEC 61158-5-2, 6.2.3.2.1.7 for a description of the Transmission Trigger and Inactivity/Watchdog timers.

The resolution of this attribute is in milliseconds. A request to configure this attribute can result in the specification of a time value that a product cannot meet. In addition to performing product specific range checking when a request to modify this attribute is received, the following steps are performed:

- If the specified value is not equal to an increment of the available clock resolution, then the value is rounded up to the next serviceable value. For example: a Set\_Attribute\_Single request is received specifying the value 5 for the **expected\_packet\_rate** attribute and the product provides a 10 millisecond resolution on timers. In this case the product would load the value 10 into the **expected\_packet\_rate** attribute.
- The value that is actually loaded into the **expected\_packet\_rate** attribute is reported in the Service Data Field of a Set\_Attribute\_Single response message associated with a request to modify this attribute.
- If the requested value is equal to an increment of the clock resolution, then the requested value is loaded into the **expected\_packet\_rate** and reported in the response. For example: if the value 100 is requested and the clock resolution is 10 milliseconds, then a value of 100 is loaded.
- When a Connection object is in the **Established** state, any modifications to the **expected\_packet\_rate** attribute have immediate effect on the Inactivity/Watchdog Timer. The following steps are performed by a Connection object in the **Established** state when a request is received to modify the **expected\_packet\_rate** attribute:
  - the current Inactivity/Watchdog Timer is canceled,
  - a new Inactivity/Watchdog Timer is activated based on the new value in the **expected\_packet\_rate** attribute.

This attribute defaults to 2500 (2 500 ms) within Explicit Messaging Connections, and to zero (0) within an I/O Connection.

#### ***MSG\_produced\_connection\_id***

This attribute is Required, unless otherwise specified in the network communication profile of the subnet. It contains the Connection ID, which identifies messages to be sent across this connection (if any).

#### ***MSG\_consumed\_connection\_id***

This attribute is Required, unless otherwise specified in the network communication profile of the subnet. It contains the Connection ID, which identifies messages to be received across this connection (if any).

#### ***Watchdog\_timeout\_action***

This attribute defines the action the Connection object should perform when the Inactivity/Watchdog Timer expires. Table 166 defines the specifics of this attribute.

**Table 166 – Values for the watchdog\_timeout\_action**

| Value       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x00        | <i>Transition to Timed Out.</i> The Connection transitions to the Timed Out state and remains in this state until it is Reset or Deleted. The command to Reset or Delete could come from an internal source (e.g., an application object) or could come from the network (e.g., a configuration tool). This is the default value for this attribute with respect to I/O Connections. This value is invalid for Explicit Messaging Connections. |
| 0x01        | <i>Auto Delete.</i> The Connection Class automatically deletes the Connection if it experiences an Inactivity/Watchdog timeout. This is the default value for this attribute with respect to Explicit Messaging Connections.                                                                                                                                                                                                                   |
| 0x02        | <i>Auto Reset.</i> The Connection remains in the Established state and immediately restarts the Inactivity/Watchdog timer. This value is invalid for Explicit Messaging Connections.                                                                                                                                                                                                                                                           |
| 0x03        | <i>Deferred Delete.</i> This value is only used by a specific Type 2 network communication profile.                                                                                                                                                                                                                                                                                                                                            |
| 0x04 - 0xFF | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                       |

***Produced\_connection\_path\_length***

Specifies the number of octets of information within the **produced\_connection\_path** attribute. This is automatically initialized when the **produced\_connection\_path** attribute is configured. This attribute defaults to the value zero (0).

***Produced\_connection\_path***

The **produced\_connection\_path** attribute is made up of a octet stream which defines the application object(s) whose data is to be produced by this Connection object. The format of this octet stream is specified in 4.1.9. This attribute defaults to being empty upon instantiation of the Connection. It remains empty within Explicit Messaging Connections and within Connection objects that do not produce.

***Consumed\_connection\_path\_length***

Specifies the number of octets of information within the **consumed\_connection\_path** attribute. This is automatically initialized when the **consumed\_connection\_path** attribute is configured. This attribute defaults to the value zero (0).

***Consumed\_connection\_path***

The **consumed\_connection\_path** attribute is made up of an octet stream which defines the application object(s) that are to receive the data consumed by this Connection object. The format of this octet stream is specified in 4.1.9. This attribute defaults to being empty upon instantiation of the Connection. It remains empty within Explicit Messaging Connections and within Connection objects that do not consume.

***Production\_inhibit\_time***

This attribute is used to configure the minimum delay time between new data production. This is required for all I/O Client connections, except those with a production trigger of Cyclic. The Set\_Attribute\_Single service shall be supported when this attribute is implemented. A value of zero (the default value for this attribute) indicates no inhibit time.

The resolution of this attribute is in milliseconds. A request to configure this attribute can result in the specification of time value that a product cannot meet. In addition to performing product specific range checking when a request to modify this attribute is received, the following steps are performed:

- If the specified value is not equal to an increment of the available clock resolution, then the value is rounded up to the next serviceable value. For example: a Set\_Attribute\_Single request is received specifying the value 5 for the **production\_inhibit\_time** attribute and the product provides a 10 millisecond resolution on times. In this case the product would load the value 10 into the **production\_inhibit\_time** attribute.
- The value that is actually loaded into the **production\_inhibit\_time** attribute is reported in the Service Data Field of a Set\_Attribute\_Single response message associated with a request to modify this attribute.
- If the requested value is equal to an increment of the clock resolution, then the requested value is loaded into the **production\_inhibit\_time** and reported in the response. For example: the value 100 is requested and the clock resolution is 10 milliseconds.

The Production Inhibit Timer shall be restarted each time new data production occurs. Data produced due to the expiration of the transmission trigger timer shall not result in the Production Inhibit Timer being restarted.

When a Connection object is in the **Established** state, any modifications to the **production\_inhibit\_time** attribute have no effect on a currently running Production Inhibit Timer. The new production\_inhibit\_time value is loaded into the Production Inhibit Timer on the following new data production.

When the apply\_attributes service is received, the **production\_inhibit\_time** shall be verified against the **expected\_packet\_rate** attribute. If the **expected\_packet\_rate** value is greater than zero, but less than the **production\_inhibit\_time** value, then an error shall be returned. In this case, where two attribute values conflict use the **production\_inhibit\_time attribute** ID as the extended error code returned in the error response.

#### **Connection\_timeout\_multiplier**

The Connection\_timeout\_multiplier specifies the multiplier applied to the expected packet rate to obtain the value used by the Inactivity/Watchdog Timer. See the Connection Timeout Multiplier parameter description within the Connection Manager object Specific Service Parameters for the enumerated values of this attribute. The default value for this attribute is zero (specifying a multiplier of 4).

#### **Connection\_binding\_list**

The Connection\_binding\_list attribute identifies connection instances that are bound to this connection. The attribute structure provides a 16 bit count of bound connections, followed by a list of the bound instances. This attribute shall be supported if the Connection\_Bind service is supported.

### **4.1.8.9.2 Common services**

#### **4.1.8.9.2.1 Apply\_Attributes**

The Apply\_Attributes common service shall have the object-specific response parameters specified in Table 168.

**Table 167 – Object-specific response parameters for Apply\_Attributes**

| Name                   | Data type | Semantics of values                                                                                                                         |
|------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Produced Connection ID | UINT      | Contains the value within the Connection Instance's produced_connection_id attribute for the relevant Type 2 network communication profile. |
| Consumed Connection ID | UINT      | Contains the value within the Connection Instance's consumed_connection_id attribute for the relevant Type 2 network communication profile. |

**4.1.8.9.2.2 Set\_Attribute\_Single**

The Set\_Attribute\_Single common service associated with the modification of the expected\_packet\_rate attribute shall have the object-specific response parameters specified in Table 168.

**Table 168 – Object-specific response parameter for Set\_Attribute\_Single**

| Name                 | Data type | Semantics of values                                                                        |
|----------------------|-----------|--------------------------------------------------------------------------------------------|
| Expected Packet Rate | UINT      | Contains the actual value within the Connection Instance's expected_packet_rate attribute. |

**4.1.8.9.3 Object specific services****4.1.8.9.3.1 Connection\_Bind Service**

The Connection\_Bind\_RequestPDU body shall be as specified in Table 169.

**Table 169 – Structure of Connection\_Bind\_RequestPDU body**

| Name            | Data type                 | Semantics of values                                 |
|-----------------|---------------------------|-----------------------------------------------------|
| Bound Instances | STRUCT of<br>UINT<br>UINT | Instance numbers of Connection objects to be bound. |

The Connection\_Bind\_ResponsePDU body is empty.

The service response may return a status from the list of status codes in Table 170.

**Table 170 – Object specific status for Connection\_Bind service**

| General Status Code | Extended Status Code | Status Description                                                                                              |
|---------------------|----------------------|-----------------------------------------------------------------------------------------------------------------|
| 0x02                | 0x01                 | One or both of the connection instances is non-existent.                                                        |
| 0x02                | 0x02                 | The connection class and/or instance is out of resources to bind instances.                                     |
| 0x0C                | 0x01                 | Both of the connection instances are existent, but at least one is not in the Established state.                |
| 0x20                | 0x01                 | Both connection instances are the same value.                                                                   |
| 0xD0                | 0x01                 | One or both of the connection instances is not a dynamically created I/O connection.                            |
| 0xD0                | 0x02                 | One or both of the connection instances were created internally and the device is not allowing a binding to it. |

#### 4.1.8.9.3.2 Producing\_Application\_Lookup\_Service

The Producing\_Application\_Lookup\_RequestPDU body shall be as specified in Table 171.

**Table 171 – Structure of Producing\_Application\_Lookup\_RequestPDU body**

| Name                       | Data type    | Semantics of values                                                                                                                                           |
|----------------------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Producing Application Path | Packed EPATH | Connection path of producing application to be searched for within connections in the Established state. The path shall be a single Logical or Symbolic path. |

The Producing\_Application\_Lookup\_ResponsePDU body shall be as specified in Table 172.

**Table 172 – Structure of Producing\_Application\_Lookup\_ResponsePDU body**

| Name                     | Data type         | Semantics of values                                                                                           |
|--------------------------|-------------------|---------------------------------------------------------------------------------------------------------------|
| Instance Count           | UINT              | Number of Instances returned in the Connection Instance List parameter.                                       |
| Connection Instance List | Struct of<br>UINT | List of instance numbers of connection producing the data from the requested connection path within the node. |

The service response may return a status from the list of status codes in Table 173.

**Table 173 – Producing\_Application\_Lookup\_Service status codes**

| General Status Code | Extended Status Code | Status Description                                                                     |
|---------------------|----------------------|----------------------------------------------------------------------------------------|
| 0x02                | 0x01                 | The connection path was not found in any connection instance in the Established state. |

#### 4.1.8.9.3.3 SafetyOpen\_Service

See IEC 61784-3-2 for the definition of this service.

#### 4.1.8.9.3.4 SafetyClose\_Service

See IEC 61784-3-2 for the definition of this service.

### 4.1.9 Message and connection paths

#### 4.1.9.1 Contents

The path shall specify the object element that is either the target of a connection request, or the destination of a message request. The path shall contain multiple segments which can indicate the route to the next node (in the case of multiple links), what to connect to or where to send a message in the target device. Each segment shall be comprised of a segment descriptor octet which specifies the segment type and format, and segment information whose size is dependent on the segment type/format.

A path has a data type EPATH (Packed or Padded). The two types of path shall be

- padded paths (indicated as data type Padded EPATH);
- packed paths (indicated as data type Packed EPATH).

These two path formats (padded and packed) shall not be interchangeable. Usage of padded versus packed shall be specified, depending on the context of use.

Each segment of a padded path shall be 16 bit word aligned. If a pad octet is required to achieve the alignment, then the segment shall specify the location of the pad octet. A packed path shall not contain any pad octets.

The possible segment types shall be as follows:

Port segment ( where );  
 Logical segment ( what );  
 Network segment ( when or how );  
 Symbolic segment;  
 Data segment.

#### 4.1.9.2 Segment type

Each segment of the path shall contain a segment type/format octet to indicate how the segment is to be interpreted. The segment type/format is contained in the first octet of the segment. The bits of the first octet of a path segment shall be numbered 0 to 7, where bit 0 shall be the least significant bit of the octet. Bits of the segment type are defined in Figure 9.

| Segment Type |   |   | Segment Format                     |   |   |   |   |
|--------------|---|---|------------------------------------|---|---|---|---|
| 7            | 6 | 5 | 4                                  | 3 | 2 | 1 | 0 |
|              |   |   |                                    |   |   |   |   |
| 0            | 0 | 0 | Port Segment                       |   |   |   |   |
| 0            | 0 | 1 | Logical Segment                    |   |   |   |   |
| 0            | 1 | 0 | Network Segment                    |   |   |   |   |
| 0            | 1 | 1 | Symbolic Segment                   |   |   |   |   |
| 1            | 0 | 0 | Data Segment                       |   |   |   |   |
| 1            | 0 | 1 | Data Type (constructed, see 5.2.3) |   |   |   |   |
| 1            | 1 | 0 | Data Type (elementary, see 5.2.2)  |   |   |   |   |
| 1            | 1 | 1 | Reserved for future use            |   |   |   |   |

**Figure 9 – Segment type**

The meaning of the Segment Format bits is based on the specified Segment Type.

#### 4.1.9.3 Port segment

The port segment shall indicate

- communication port through which to leave the node (expressed as a 16-bit number);
- link address of the next device in the path.

Figure 10 shows the overall structure of the port segment.

Bits 5 to 7 of the first octet shall be zero to indicate the port segment type.

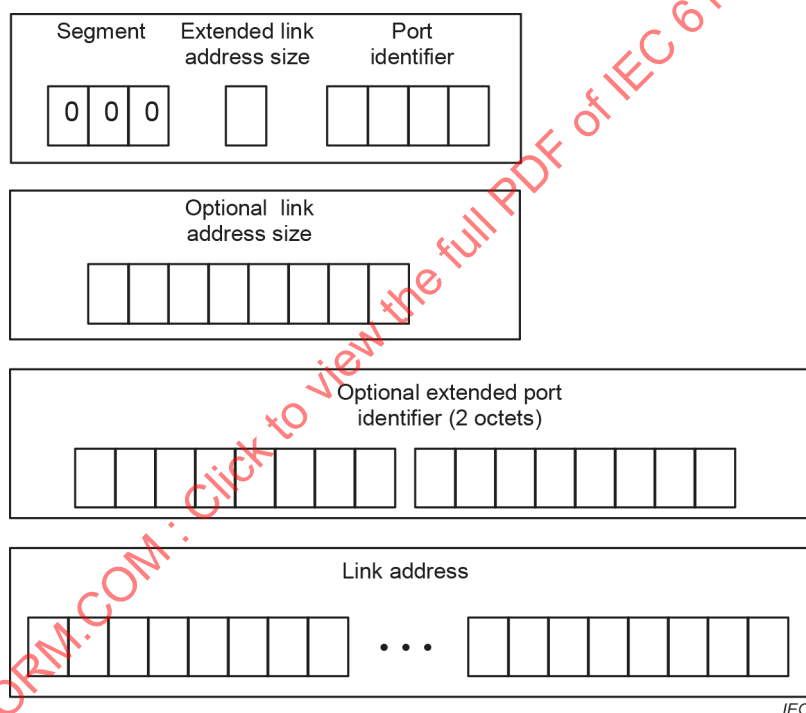
Bit 4, the extended link address size bit, shall be set to 0 if the link address is one octet. Bit 4 shall be set to 1 if the link address is larger than one octet. If the link address is larger than one octet, its size in octets shall be in the second octet of the port segment.

To support the Port Object instance attribute 7 (Port Number and Node Address, see IEC 61158-4-2, 7.11.4.6), a port segment with no link address can be specified by setting the Extended Link Address Size bit (Bit 4 set to 1) and setting the Optional Link Address Size octet to 0.

Bits 0 to 3, the port identifier, shall indicate through which port to leave the node. The port identifier shall specify a port number or an escape to an extended port identifier when the module can support more than 14 ports. Port number 0 shall be reserved. Port number 1 shall only be used to represent a back-plane port. If the port identifier is 15, then a 16-bit field, called the extended port identifier, shall be the next part of the port segment (following the optional link address size if present); otherwise, the value of the port identifier shall be the port number.

The port number shall be followed by a link address whose format depends on the type of network to which the port identifier refers. If the link address is greater than one octet, then it shall be padded so that the entire port segment is an even number of octets. The pad octet shall be set to 0 and shall not be included in the link address size.

NOTE 1 For the common port types, the link address is a single octet. Other port types, such as TCP/IP encapsulation, can use a larger link address (see 4.3 and Clause 11).



NOTE The bit representation is from high bit to low bit, left to right. The octet representation is from low octet to high octet, top to bottom and left to right.

**Figure 10 – Port segment**

The Extended Port Identifier format of the Port Segment shall only be used when there are more than 14 ports possible on the device. The Port Segment shall always be packed into the smallest Port Segment format possible with respect to the optional fields. Examples of possible port segments are as shown in Table 174 (values are hexadecimal).

**Table 174 – Possible port segment examples**

| Port segment contents                                                            | Notes                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [02][06]                                                                         | Segment type = port segment, port number = 2, link address = 6                                                                                                                                                             |
| [0F][12][00][01]                                                                 | Segment type = port segment. Port identifier is 15, indicating the port number is specified in the next 16 bit field [12][00] (18 decimal). Link address = 1                                                               |
| [15][0F][31][33][30][2E]<br>[31][35][31][2E][31][33]<br>[37][2E][31][30][35][00] | Segment type = port segment. Multi-octet address for TCP Port 5, link address 130.151.137.105 (IP address). The address is defined as a character array, length of 15 octets. The last octet in the segment is a pad octet |

The link address portion of a TCP/IP connection path segment shall be encoded within the port segment as a string of ASCII characters. The string shall be one of the following forms:

- IP address in dotted decimal notation, for example "130.151.132.55" (see IETF RFC 1117 for the format of IP addresses);
- IP address in dotted decimal notation, followed by a ":" separator, followed by the TCP port number to be used at the specified IP address;
- host name, for example "plc.type2.org". The host name shall be resolved via a DNS request to a name server (see IETF RFC 1035 for information on host names and name resolution);
- host name, followed by a ":" separator, followed by the TCP port number to be used at the specified host.

The client generating the path shall specify all numbers supplied as parts in IP dotted decimal notation as decimal values with no leading "0x" or "0". Server handling of leading "0x" or "0" is implementation dependent.

The port number shall be represented in either hex or decimal. Hex shall be indicated by a leading "0x". When a port number is specified, it shall be used rather than the standard port number used for the encapsulation protocol (0xAF12). Only port 0xAF12 is guaranteed to be available in a Type 2 Ethernet compliant device.

Other TCP port numbers may be implemented; however, this specification does not provide a mechanism to determine which TCP port numbers are supported by a device. The use of other TCP port numbers is therefore discouraged.

NOTE 2 The guaranteed TCP port number, 0xAF12, has been reserved with the Internet Assigned Numbers Authority (IANA) for use by the encapsulation protocol.

Since port segments shall be word-aligned, a pad octet can be required at the end of the string. The pad octet shall be 0x00, and shall not be counted in the Optional Address Size field of the port segment.

NOTE 3 Examples of port segments for TCP/IP are shown in Table 175 below (values are hexadecimal).

**Table 175 – TCP/IP link address examples**

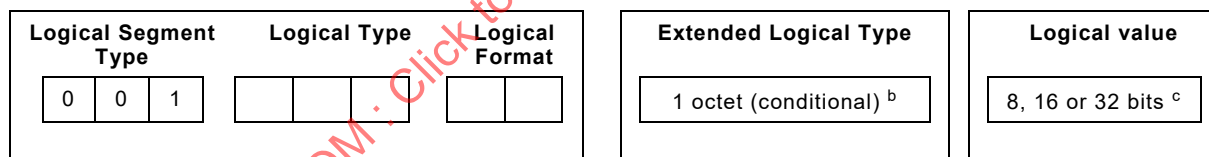
| Port segment                                                                                                         | IP address            | Notes                                                            |
|----------------------------------------------------------------------------------------------------------------------|-----------------------|------------------------------------------------------------------|
| [12][0D]<br>[31][33][30][2E]<br>[31][35][31][2E]<br>[31][33][32][2E][31][00]                                         | 130.151.132.1         | Multi-octet address for port 2, 13 octet string plus a pad octet |
| [13][0C]<br>[70][6C][63][2E]<br>[74][79][70][32][2E]<br>[6F][72][67]                                                 | plc.typ2.org          | Multi-octet address for port 3, 12 octet string, no pad octet    |
| [16][15]<br>[31][33][30][2E]<br>[31][35][31][2E]<br>[31][33][32][2E]<br>[35][35][3A]<br>[30][78][33][32][31][30][00] | 130.151.132.55:0x3210 | Multi-octet address for port 6, 21 octet string plus a pad octet |
| [15][11]<br>[70][6C][63][2E]<br>[74][79][70][32][2E]<br>[6F][72][67][3A]<br>[39][38][37][36][00]                     | plc.typ2.org:9876     | Multi-octet address port 5, 17 octet string plus a pad octet     |

#### 4.1.9.4 Logical segment

##### 4.1.9.4.1 Logical segment structure

The logical segment selects a particular addressable entity within a device (for example, object class, object instance, and object attribute). When the logical segment is included within a Packed Path, the Logical Value shall be appended to the segment type octet with no pad in between.

Figure 11 specifies the encoding of a logical segment.



|                         |   |   |   |   |   |                         |
|-------------------------|---|---|---|---|---|-------------------------|
| Class ID                | 0 | 0 | 0 | 0 | 0 | 8-bit logical value     |
| Instance ID             | 0 | 0 | 1 | 0 | 1 | 16-bit logical value    |
| Member ID               | 0 | 1 | 0 | 1 | 0 | 32-bit logical value    |
| Connection Point        | 0 | 1 | 1 | 1 | 1 | Reserved for future use |
| Attribute ID            | 1 | 0 | 0 |   |   |                         |
| Special <sup>a</sup>    | 1 | 0 | 1 |   |   |                         |
| Service ID <sup>a</sup> | 1 | 1 | 0 |   |   |                         |
| Extended Logical        | 1 | 1 | 1 |   |   |                         |

<sup>a</sup> The Special and Service ID Logical Types do not use the logical addressing definition for the Logical Format.

<sup>b</sup> Required when Logical Type is Extended Logical (1 1 1), not present otherwise. The Extended Logical Type values are defined in Table 176.

<sup>c</sup> Depends on value of Logical Format.

**Figure 11 – Logical segment encoding**

The Extended Logical Types are specified in Table 176.

**Table 176 – Extended Logical Type**

| Value    | Description             |
|----------|-------------------------|
| 0        | Reserved                |
| 1        | Array Index             |
| 2        | Indirect Array Index    |
| 3        | Bit Index               |
| 4        | Indirect Bit Index      |
| 5        | Structure Member Number |
| 6        | Structure Member Handle |
| 7 to 255 | Reserved                |

The 8-bit and 16-bit logical address formats are allowed for use with all Logical Types.

The 32-bit logical address format is only allowed for the logical Instance ID, Member ID, Connection Point, Array Index, Bit Index, Structure Member Number and Structure Member Handle types. It is not allowed for any other Logical Type (reserved for future use).

The Class ID specifies a type of object within a device. The Instance ID is an integer assigned to an object when it is created. Instance 0, called the class instance, specifies the class itself. The attributes of an object are an enumerated list of externally visible values. The attribute ID specifies which attribute of an object is selected. Complex attributes are composed of members. The member ID specifies which member of the attribute.

When an extended logical segment is included within a Padded Path and the Logical Format is 8 bit, a pad octet shall be added after the Logical Value (the 16-bit and 32-bit formats are identical to the Packed Path) and shall be set to 0. For all other logical segments, when included within a Padded Path, the 16-bit and 32-bit logical formats shall have a pad inserted between the segment type octet and the Logical Value (the 8-bit format is identical to the Packed Path). The pad octet shall be set to zero.

When an Extended Logic Segment is specified the Extended Logical Segment Value octet shall be included between the Segment Type octet and the Logical Value.

An Array Index Extended Logical Segment is used to specify the 0-based index into the preceding item in the EPATH which shall be an array.

An Indirect Array Index Extended Logical Segment is used to specify the start of a nested application path, which resolves to a 0-based index into the preceding item in the EPATH which shall be an array. The Indirect Array Index Logical Segment Logical Value is the size of the nested application path, in 16-bit words.

A Bit Index Extended Logical Segment is used to specify the 0-based bit number into the preceding item in the EPATH (there is no restriction on type of the preceding item).

An Indirect Bit Index Extended Logical Segment is used to specify the start of a nested application path, which resolves to a 0-based bit number into the preceding item in the EPATH (there is no restriction on type of the preceding item). The Indirect Bit Index Logical Segment Logical Value is the size of the nested application path, in 16-bit words.

A Structure Member Number Extended Logical Segment is used to specify the 1-based member number into the preceding item in the EPATH which shall be a structure. The member number specifies the structure member, where the structure members are numbered starting with 1.

A Structure Member Handle Extended Logical Segment is used to specify the member handle into the preceding item in the EPATH which shall be a structure. The handle is a value that uniquely identifies a member and is provided by the target device when reporting constructed data type information using the Formal Encoding with Handles specified in 5.2.3.2.5.

The Connection Point Logical Type provides additional addressing capabilities beyond the standard Class ID/Instance ID/Attribute ID/Member ID object address. Object classes shall define when and how this addressing component is utilized (for example, a sub-instance of an Assembly object).

NOTE IEC 61158-5-2 describes the object model including the definition of class, instance, attribute, connection point, and members of a library of common objects.

The Service ID Logical Type has the following definition for the Logical Format:

- 0 0 8-Bit Service ID Segment (0x38)
- 0 1 Reserved for future use (0x39)
- 1 0 Reserved for future use (0x3A)
- 1 1 Reserved for future use (0x3B)

The Special Logical Type has the following definition for the Logical Format:

- 0 0 Electronic Key Segment (0x34)
- 0 1 Reserved for future use (0x35)
- 1 0 Reserved for future use (0x36)
- 1 1 Reserved for future use (0x37)

#### 4.1.9.4.2 Electronic Key Segment

The electronic key segment shall be used to verify/identify a device. The key may be included as the first logical segment in a connection path and shall be checked by the Message Router of the receiving node against the values contained in instance 1 of its Identity object before any additional address checks are made. On a multi-hop message, sent to a remote link via routers, multiple electronic key segments may appear. The format of the electronic key segment shall be as shown in Table 177.

**Table 177 – Electronic key segment format**

| Parameter           | Format          | Value                                                                                                                                              |
|---------------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| segment_type        | USINT           | always 0x34                                                                                                                                        |
| electronic_key_type | USINT           | 0 to 3 – reserved<br>4 – REQUIRED (see Key Format in Table 178)<br>5 – OPTIONAL (see Serial Number Key Format in Table 179)<br>6 to 255 – reserved |
| electronic_key_data | ARRAY of octets | Depends on key format used                                                                                                                         |

The segment\_type for a key segment shall be 0x34. The electronic\_key\_type shall determine the format of a specific key segment: If electronic\_key\_type is equal to 0x04, the format of electronic\_key\_data is specified in Table 178. If electronic\_key\_type is equal to 0x05, the format of electronic\_key\_data is specified in Table 179. All other values for the electronic\_key\_type shall be reserved.

**Table 178 – Key Format Table (key type 4)**

| Parameter           | Format | Value                                                               |
|---------------------|--------|---------------------------------------------------------------------|
| vendor_ID           | UINT   | these correspond to the values in instance 1 of the identity object |
| product_type        | UINT   |                                                                     |
| product_code        | UINT   |                                                                     |
| major_revision: 7   | USINT  |                                                                     |
| compatible_match: 1 | USINT  |                                                                     |
| minor_revision      | USINT  |                                                                     |

**Table 179 – Serial Number Key Format Table (key type 5)**

| Parameter           | Format | Value                                                               |
|---------------------|--------|---------------------------------------------------------------------|
| vendor_ID           | UINT   | these correspond to the values in instance 1 of the identity object |
| product_type        | UINT   |                                                                     |
| product_code        | UINT   |                                                                     |
| major_revision: 7   | USINT  |                                                                     |
| compatible_match: 1 | USINT  |                                                                     |
| minor_revision      | USINT  |                                                                     |
| serial_number       | UDINT  |                                                                     |

The `vendor_ID` shall specify the device vendor, or zero if no specific vendor is required.

The `product_type` shall specify a class of products such as digital input or analogue outputs. The `product_type` shall be ignored when it is set to zero. The `product_code` shall be vendor specific with each vendor having a unique code. The `product_code` shall be ignored when set to zero.

The `major_revision` and `compatible_match` fields shall be contained in the least significant 7 bits and most significant bit of the same octet, respectively.

The `major_revision` shall specify the functionality level of a device. The `major_revision` shall be ignored when set to zero.

The `minor_revision` shall specify the revision of a device that does not effect network-visible behavior. A value of zero shall specify that the originator of the request accepts any minor revision.

The `serial_number` shall specify the serial number of a device. When used in conjunction with the `vendor_ID`, it forms a unique identifier for each device on any Type 2 network. It shall not be set to zero.

If the `serial_number` received in the request is 0 or does not match the `serial_number` of the device, then the request shall be rejected. If the request was directed at the Message Router object, then general status 0x25, extended status 0x013A shall be returned (see Table 205). If the request was directed at the Connection Manager object, then general status 0x01, extended status 0x013A shall be returned (see Table 203).

The `compatible_match` shall modify the key matching function.

If the bit is clear (= 0), then any non-zero `vendor_ID`, `product_type`, `product_code`, `major_revision`, and `minor_revision` shall match exactly.

If the bit is set (=1), any key which a device can emulate may be accepted.

The `vendor_ID`, `product_type`, `product_code`, and `revision` are used to identify the device the recipient is being requested to emulate; therefore 0 is an invalid value for the `vendor_ID`, `product_type`, `product_code`, `major_revision` and `minor_revision` fields.

If the `vendor_ID`, `product_type`, `product_code`, `major_revision` and/or `minor_revision` is 0 or the device cannot emulate the requested device, the request shall be rejected. If the compatible key was included in a request directed at the Message Router object, then one of the errors in Table 205 shall be returned. If the request was directed at the Connection Manager object, then one of the key-specific errors (general status 0x01 and extended status 0x0114 to 0x0116) in Table 203 shall be returned.

EXCEPTION: Device Type 0 is allowed if the recipient can emulate a device that reports the deprecated Generic Device Type Number.

NOTE IEC 61158-5-2 describes the Identity object and the rules for updating the revision fields.

#### 4.1.9.4.3 Logical segments examples

Table 180 shows examples of logical segments.

**Table 180 – Logical segments examples**

| First octet | Logical segment name    | Padded format examples (as transmitted)                      | Packed format examples (as transmitted)                      | Notes                                                                                                                                                  |
|-------------|-------------------------|--------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x20        | 8-bit class             | [20][04]                                                     | [20][04]                                                     | class 0x04 (Assembly object)                                                                                                                           |
| 0x21        | 16-bit class            | [21][00][34][12]                                             | [21][34][12]                                                 | class 0x1234                                                                                                                                           |
| 0x24        | 8-bit instance          | [24][04]                                                     | [24][04]                                                     | instance 0x04                                                                                                                                          |
| 0x25        | 16-bit instance         | [25][00][34][12]                                             | [25][34][12]                                                 | instance 0x1234                                                                                                                                        |
| 0x28        | 8-bit member            | [28][04]                                                     | [28][04]                                                     | member 0x04                                                                                                                                            |
| 0x29        | 16-bit member           | [29][00][34][12]                                             | [29][34][12]                                                 | member 0x1234                                                                                                                                          |
| 0x2A        | 32-bit member           |                                                              |                                                              |                                                                                                                                                        |
| 0x2C        | 8-bit connection point  | [2C][04]                                                     | [2C][04]                                                     | connection point 0x04                                                                                                                                  |
| 0x2D        | 16-bit connection point | [2D][00][34][12]                                             | [2D][34][12]                                                 | connection point 0x1234                                                                                                                                |
| 0x30        | 8-bit attribute         | [30][04]                                                     | [30][04]                                                     | attribute 0x04                                                                                                                                         |
| 0x31        | 16-bit attribute        | [31][00][34][12]                                             | [31][34][12]                                                 | attribute 0x1234                                                                                                                                       |
| 0x34        | electronic key          | [34][04]<br>[12][00]<br>[EF][BE]<br>[0E][0B]<br>[83]<br>[01] | [34][04]<br>[12][00]<br>[EF][BE]<br>[0E][0B]<br>[83]<br>[01] | vendor ID = 0x0012<br>product type = 0xBEEF<br>product code = 0x0B0E<br>major revision = 0x03<br>minor revision = 0x01<br>accept compatible keys = yes |

#### 4.1.9.5 Network segment

##### 4.1.9.5.1 Network segment structure

The segment type (first octet) of a network segment shall be in the range 0x40 through 0x5F as shown in Table 181 (the most significant bits shall be 010). The network segment shall be used to specify network parameters which could be required by a node to transmit a message across a network. The network segment shall immediately precede the port segment of the device to which it applies. In other words, the network segment shall be the first item in the path that the device receives.

**Table 181 – Network segments**

| First octet  | Network segment name                    |
|--------------|-----------------------------------------|
| 0x40         | reserved                                |
| 0x41         | schedule segment                        |
| 0x42         | fixed tag segment                       |
| 0x43         | production inhibit time in milliseconds |
| 0x44 to 0x4F | reserved                                |
| 0x50         | safety segment                          |
| 0x51         | production inhibit time in microseconds |
| 0x52         | user authentication session ID          |
| 0x53 to 0x5E | reserved                                |
| 0x5F         | extended network segment                |

##### 4.1.9.5.2 Schedule Segment

This segment is reserved for the Type 2 network communication profile using the data link layer specified in IEC 61158-3-2.

The segment type of the schedule network segment shall be 0x41. The schedule network segment shall specify

- a multiplier that, when multiplied by the NUT, gives the CM\_API (actual packet interval) of the scheduled transport;
- a phase that determines on which values of the `Moderator.interval_count` to transmit.

NOTE 1 The data link layer defines the `Moderator.interval_count`, see IEC 61158-4-2.

This segment shall be included in the path to a device so that each intermediary node can schedule link transmit time for subsequent scheduled traffic. The multiplier shall be one of 1, 2, 4, 8, 16, 32, 64 or 128. The phase shall be in the range 0 through (multiplier – 1). The second octet of the schedule network segment shall encode both the multiple and phase by adding them together.

NOTE 2 If a transport produces every 64<sup>th</sup> NUT starting on `interval_count` = 17, then the encoded value is 17 + 64 = 81.

An encoded value of zero shall specify that the transport has not yet been given permission to use the scheduled priority. If a node receives a request for a scheduled connection in which either no schedule network segment exists, or the schedule network segment exists but the encoded value is 0, the node shall return general status = 0x01, and extended status = 0x0317.

NOTE 3 A connection originator is given permission to use the scheduled priority through the Scheduling object (IEC 61158-4-2).

The number of required schedule segments for intermediate and target nodes is defined in the Type 2 network communication profile.

#### 4.1.9.5.3 Fixed Tag Segment

This segment is reserved for the Type 2 network communication profile using the data link layer specified in IEC 61158-3-2.

The segment type of the fixed tag network segment shall be 0x42. The fixed tag network segment shall specify the fixed tag which is to be used when sending an unconnected message. This segment subtype shall precede the port segment within the path. If the fixed tag segment is not present, then the default fixed tag shall be used.

NOTE The fixed tag segment can be used within the path of the Unconnected\_Send service (see 4.1.5.6) of the Connection Manager to send requests to the Keeper object via the Management UCMM (see IEC 61158-5-2).

#### 4.1.9.5.4 Production Inhibit Time Network Segments

##### 4.1.9.5.4.1 General

The production inhibit time network segments shall specify the minimum time between successive transmissions of new data for the specified connection. The Production Inhibit Time Network Segments only apply to Change of State or Application triggered connections. A value of zero shall indicate no production inhibit time. When a production inhibit segment is not provided during establishment of a connection, a default value of one-fourth the RPI shall be used.

The production inhibit time shall be less than or equal to the RPI. If the production inhibit time is greater than the RPI, the Forward\_Open response shall be returned with one of the following errors:

- Production Inhibit Time is greater than the RPI (status 0x01, extended status 0x11B) – recommended;
- RPI not supported (status 0x01, extended status 0x0111) – deprecated.

The production inhibit time network segment only applies to the target device.

Target devices may support neither of the Production Inhibit Time network segments. If the device supports the Production Inhibit Time in microseconds Network Segment it shall also support the Production Inhibit Time in milliseconds Network Segment.

NOTE The method for transmitting APDUs over Ethernet-TCP/IP is specified in 4.3 and Clause 11.

##### 4.1.9.5.4.2 Production Inhibit Time in milliseconds Network Segment

The Production Inhibit Time in milliseconds Network Segment specifies a time in milliseconds.

The time specified in the Production Inhibit Time in Milliseconds Network Segment corresponds to the production inhibit definition in the Connection object production\_inhibit\_time attribute (17) definition.

EXAMPLE If a production inhibit time of 10 ms is specified, new data shall be sent no sooner than 10 ms after the previous new data production.

The network segment data field shall be a single USINT indicating the production inhibit time in milliseconds. The production inhibit time range is 0 ms to 255 ms.

#### 4.1.9.5.4.3 Production Inhibit Time in microseconds Network Segment

The Production Inhibit Time in microseconds Network Segment specifies a time in microseconds.

The time specified in the Production Inhibit Time in microseconds Network Segment is used in the manner defined in the production inhibit definition in the Connection Object production\_inhibit\_time attribute (17) definition, except that the production inhibit time is in microseconds instead of milliseconds and is a UDINT instead of a USINT.

EXAMPLE If a production inhibit time of 250  $\mu$ s is specified, new data shall be sent no sooner than 250  $\mu$ s after the previous new data production.

The Network Segment Data field shall consist of:

- a USINT indicating the Number of Data Words field. The value shall be 2.
- a UDINT indicating the production inhibit time in microseconds, the production inhibit time range is 0  $\mu$ s to 4 294 967 295  $\mu$ s.

#### 4.1.9.5.5 Safety Segment

See IEC 61784-3-2 for format.

#### 4.1.9.5.6 User Authentication Session ID Network Segment

Specification of the format for the User Authentication Session ID Network Segment is outside the scope of this document.

#### 4.1.9.5.7 Extended Network Segment

The extended network segment allows for the definition of additional network segment subtypes. The first word of data is the extended network segment subtype. The structure of the extended network segment is shown in Figure 12.

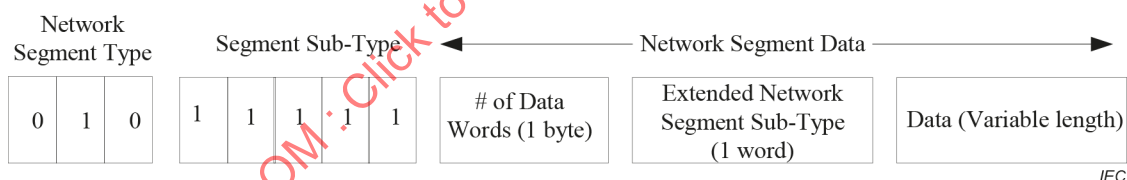


Figure 12 – Extended network segment

Each extended subtype defines the data that follows. The extended subtypes are enumerated in Table 182.

Table 182 – Extended network segment subtype definitions

| Subtype Value                                                      | Subtype Definition                              |
|--------------------------------------------------------------------|-------------------------------------------------|
| 0x0000                                                             | Open, reserved for future use                   |
| 0x0001                                                             | Participant ID for I/O aggregation <sup>a</sup> |
| 0x0002 to 0x7FFF                                                   | Open, reserved for future use                   |
| 0x8000 to 0xFFFF                                                   | Vendor specific                                 |
| <sup>a</sup> I/O aggregation is outside the scope of this document |                                                 |

#### 4.1.9.6 Symbolic segment

The range of segment types reserved for symbolic segments shall be 0x60 through 0x7F.

The symbolic segment contains an International String symbol which shall be interpreted by the device. Its format is specified in Figure 13.

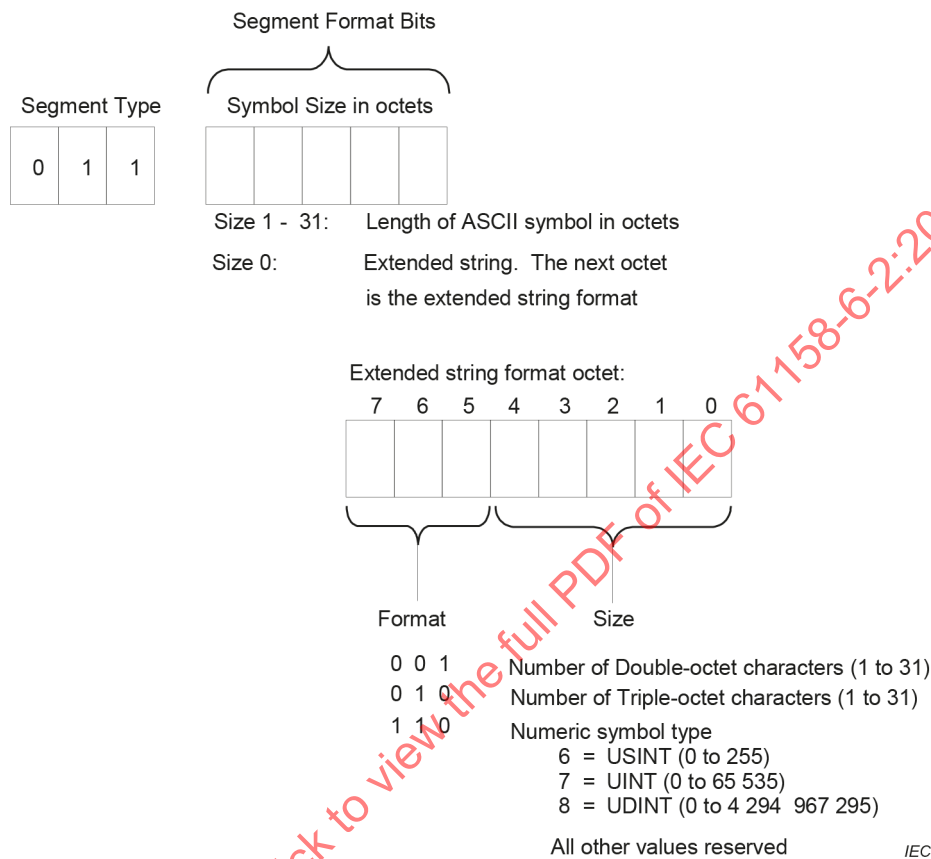


**Figure 13 – Symbolic segment encoding**

Character symbols can contain a maximum of 31 characters.

If the symbol is ASCII, double-octet or triple-octet, when comparing segments, the symbol portion shall be handled in a case insensitive manner. For instance "LS101" and "ls101" shall be considered equal.

Numeric symbols can be 8, 16, or 32 bits.

When a symbolic segment is used in a Padded EPATH, if the length of the segment is odd a pad octet of 0 shall be added at the end of the segment. The size shall not include the pad octet.

Table 183 shows examples of symbolic segments (values are hexadecimal).

**Table 183 – Symbolic segment examples**

| Symbolic Segment     | Notes                                                             |
|----------------------|-------------------------------------------------------------------|
| [65][LS101]          | ASCII Symbol. This could refer to a bit in a data structure       |
| [67][Line_23]        | ASCII Symbol. This could refer to a controller in a slot          |
| [68][Wire_off]       | ASCII Symbol. This could refer to a bit in a diagnostic structure |
| [60][22][1234][2345] | Japanese symbol                                                   |
| [60][C7][1234]       | 16 bit Numeric Symbol                                             |
| [60][C8][12345678]   | 32 bit Numeric Symbol                                             |

#### 4.1.9.7 Data segment

##### 4.1.9.7.1 Data segment purpose and structure

The data segment provides a mechanism for delivering data to an application. This may occur during connection establishment, or at any other time as defined by the application.

The segment type (first octet) of a data segment shall be in the range 0x80 through 0x9F (the three most significant bits shall be 100). This document only defines the format of the simple data segment with a segment type of 0x80; and the ANSI extended symbol segment (0x91). All other data segment types shall be reserved (0x81 through 0x90, 0x92 through 0x9F).

##### 4.1.9.7.2 Simple Data Segment

The segment type of the simple data segment shall be 0x80. The second octet shall represent the number of 16-bit words of variable length data (up to 255). Following the second octet shall be the variable length data as shown in Table 184. A path may contain more than one simple data segment.

**Table 184 – Data segment**

| Parameter    | Format | Value                                    |
|--------------|--------|------------------------------------------|
| segment_type | USINT  | always 0x80                              |
| segment_size | USINT  | size of the data[] array in 16-bit words |
| data[]       | UINT   |                                          |

The simple data segment contains data values such as parameters for the target application. The data can be configuration information for an object, additional parameters necessary for the object, or any other set of object specific information. The data segment shall not be interpreted by any other device in the path, so only the originating and target applications need agree on its contents.

##### 4.1.9.7.3 ANSI Extended Symbol Segment

The segment type of the ANSI extended symbol segment shall be 0x91. The second octet shall represent the number of characters (8-bit) in the symbol. Following the second octet shall be the variable length symbol as shown in Table 185.

The character portion of the ANSI extended symbol segment shall be handled in a case insensitive manner. For instance "start1" and "Start1" shall be considered equal.

**Table 185 – ANSI\_Extended\_Symbol segment**

| Parameter    | Format | Value                                                  |
|--------------|--------|--------------------------------------------------------|
| segment_type | USINT  | always 0x91                                            |
| symbol_size  | USINT  | size of the symbol[] array                             |
| symbol[]     | USINT  |                                                        |
| pad          | USINT  | only present if symbol_size is odd, always set to zero |

The `symbol_size` shall be the size of the `symbol[]` array in octets and shall not be zero. The `symbol_size` shall not count the `pad` octet if it is included.

#### 4.1.9.8 Segment definition hierarchy

In general, the definition of any rules related to the use of segments is defined within the Object Class and/or Device Profile. When symbolic and/or logical segments are used to construct an application path, the Backus-Naur Form definition is:

*application\_path*::= CHOICE {*symbolic\_application\_path*, *class\_application\_path*, *assembly\_class\_application\_path*}

*symbolic\_application\_path*::= *symbolic\_segment* [Connection\_Point] [*member\_specification*] [*bit\_specification*]

*symbolic\_segment*::= CHOICE {Symbolic\_Segment, ANSI\_Extended\_Symbol\_Segment}

*assembly\_class\_application\_path*::= 20 04 [*assembly\_attribute\_specification*]

*class\_application\_path*::= Class\_ID [*item\_specification*]

NOTE The Class ID cannot specify the Assembly Object (20 04).

*item\_specification*::= CHOICE {*attribute\_specification*, *connection\_point\_specification*}

*assembly\_attribute\_specification*::= CHOICE {Instance ID, Connection Point} [[Attribute ID] [*member\_specification*] [*bit\_specification*]]

*attribute\_specification*::= Instance ID [[Attribute ID] [*member\_specification*] [*bit\_specification*]]

*connection\_point\_specification*::= [Instance ID] Connection Point [*member\_specification*] [*bit\_specification*]

*member\_specification*::= CHOICE {Member\_ID, *extended\_member\_specification*}

*extended\_member\_specification*::= SEQUENCE of {Array\_Index, *indirect\_array\_specification*, Structure\_Member\_Number, Structure\_Member\_Handle}

*indirect\_array\_specification*::= Indirect\_Array\_Index *application\_path*

*bit\_specification*::= CHOICE {Bit\_Index, *indirect\_index\_bit\_specification*}

*indirect\_bit\_specification*::= Indirect\_Bit\_Index *application\_path*

A *Symbolic\_application\_path* shall resolve, at run time, to a *Class\_application\_path*.

An Indirect\_Array\_Index and Indirect\_Bit\_Index *application\_path* shall evaluate to a positive integer.

The depth within the application path need only proceed to the degree required by its application.

#### EXAMPLE

The following are examples of valid application paths.

Class ID,

Class ID, Instance ID

Class ID, Instance ID, Attribute ID

Class ID, Instance ID, Attribute ID, Member ID

Class ID, Connection Point

Class ID, Connection Point, Member ID

Class ID, Instance ID, Connection Point (see constraint a) below)

Class ID, Instance ID, Connection Point, Member ID (see constraint a) below)

Symbolic ID

Symbolic ID, Member ID

Symbolic ID, Connection Point (see constraint b) below)

Symbolic ID, Connection Point, Member ID (see constraint b) below)

Symbolic ID, Array Index, Structure Member Number, Array Index, Bit Index

Symbolic ID, Indirect\_Array\_Index, [Class\_ID, Instance\_ID, Attribute\_ID], Structure\_Member\_Number, Array\_Index, Bit\_Index

The following constraints apply:

- a) Instance ID and Connection Point for Class ID 4 (Assembly Object) shall not be used together, since for the Assembly object they are defined to be the same.
- b) If the Symbolic ID resolves to a Class ID of 4 (Assembly Object) and Instance ID then Symbolic ID and Connection Point shall not be used together, since for the Assembly object Instance and Connection Point are defined to be the same.

#### 4.1.9.9 Encoded path compression rules

When multiple encoded paths are concatenated, the delineation between paths is where a segment at a higher level in the hierarchy is encountered. Multiple encoded paths may be compacted when each path shares the same values at the higher levels in the hierarchy. Extended Logical segments shall not be used in compressed paths. When a segment is encountered which is at the same or higher level but not at the top level in the hierarchy, the preceding higher levels are used for that next encoded path.

The examples below show multiple encoded paths in the full and compacted representations.

#### EXAMPLE 1

Full: Class A, Instance A, Attribute A, Class A, Instance A, Attribute B

Compact: Class A, Instance A, Attribute A, Attribute B

#### EXAMPLE 2

Full: Class A, Instance A, Attribute A, Class A, Instance B, Attribute A

Compact: Class A, Instance A, Attribute A, Instance B, Attribute A

The following special compact format is also defined, but only when the class is not the assembly object:

Full:           Class A, Instance B, Class A, Instance B, Connection Point C, Class A, Instance B, Connection Point D, Data Segment

Compact:    Class A, Instance B, Connection Point C, Connection Point D, Data Segment

When a connection path includes a configuration or I/O path consisting of an instance of the Assembly object (class code 4) without an attribute, the data attribute (attribute 3) is assumed.

#### EXAMPLE 3

An encoded path with the contents 20 04 24 xx 24 yy 24 zz followed by a data segment is decoded as follows:

20 04 24 xx 30 03 configuration path

20 04 24 yy 30 03 consuming I/O path

20 04 24 zz 30 03 producing I/O path

### 4.1.10 Class, attribute and service codes

#### 4.1.10.1 Code ranges

##### 4.1.10.1.1 Defined ranges

There shall be three categories for address ranges of Class IDs, Attribute IDs, Connection Points and Service Codes as specified in Table 186.

**Table 186 – Addressing categories**

| This category                                                 | Refers to                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Open/Public                                                   | A value which has the same meaning for all implementers. All object classes and services defined in IEC 61158-5-2 fall under this category.                                                                                                                                    |
| Vendor-specific                                               | A range of values specific to the vendor of a device. These are used by vendors to extend their devices beyond the available <i>Open/Public</i> options. A vendor internally manages the use of values within this range. Applies to object classes, attributes, and services. |
| Object-class specific                                         | A range of values whose meaning is defined by an object class. This range applies to Service Code definitions.                                                                                                                                                                 |
| NOTE Open values are assigned by the ODVA, Inc. organization. |                                                                                                                                                                                                                                                                                |

The Class ID, Attribute ID and Service code values shall be as defined in Table 187 through Table 191.

##### 4.1.10.1.2 Class code ID ranges

The Class ID values shall be as shown in Table 187. Class Code = 0x00 is reserved and shall not be used.

**Table 187 – Class code ID ranges**

| Range (hex)                                                                 | Range (decimal) | Meaning         | Quantity |
|-----------------------------------------------------------------------------|-----------------|-----------------|----------|
| 0x00 to 0x63                                                                | 0 to 99         | Open            | 100      |
| 0x64 to 0xC7                                                                | 100 to 199      | Vendor Specific | 100      |
| 0xC8 to 0xEF                                                                | 200 to 239      | Reserved        | 40       |
| 0xF0 to 0x2FF                                                               | 240 to 767      | Open            | 528      |
| 0x300 to 0x4FF                                                              | 768 to 1 279    | Vendor Specific | 512      |
| 0x500 to 0xFFFF                                                             | 1 280 to 65 535 | Reserved        | 64 256   |
| NOTE Reserved ranges can be further defined by the ODVA, Inc. organization. |                 |                 |          |

**4.1.10.1.3 Attribute ID ranges**

The Class Attribute ID values shall be as shown in Table 188.

**Table 188 – Class Attribute ID ranges**

| Range (hex)                                                                 | Range (decimal) | Meaning                                                                   | Quantity |
|-----------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------|----------|
| 0x00                                                                        | 0               | Open                                                                      | 1        |
| 0x01 to 0x07                                                                | 1 to 7          | Reserved Class Attributes for all object class definitions, see Table 193 | 7        |
| 0x08 to 0x63                                                                | 8 to 99         | Object Specific                                                           | 92       |
| 0x64 to 0xC7                                                                | 100 to 199      | Vendor Specific                                                           | 100      |
| 0xC8 to 0xE3                                                                | 200 to 227      | Reserved Class Attributes for all object class definitions, see Table 193 | 28       |
| 0xE4 to 0xFF                                                                | 228 to 255      | Reserved                                                                  | 28       |
| 0x100 to 0x2FF                                                              | 256 to 767      | Object Specific                                                           | 512      |
| 0x300 to 0x4FF                                                              | 768 to 1 279    | Vendor Specific                                                           | 512      |
| 0x500 to 0x8FF                                                              | 1 280 to 2 303  | Open                                                                      | 1 024    |
| 0x900 to 0xCFF                                                              | 2 304 to 3 327  | Vendor Specific                                                           | 1 024    |
| 0xD00 to 0xFFFF                                                             | 3 328 to 65 535 | Reserved                                                                  | 62 208   |
| NOTE Reserved ranges can be further defined by the ODVA, Inc. organization. |                 |                                                                           |          |

The Instance Attribute ID values shall be as shown in Table 189.

**Table 189 – Instance Attribute ID ranges**

| Range (hex)                                                                 | Range (decimal) | Meaning         | Quantity |
|-----------------------------------------------------------------------------|-----------------|-----------------|----------|
| 0x00 to 0x63                                                                | 00 to 99        | Open            | 100      |
| 0x64 to 0xC7                                                                | 100 to 199      | Vendor Specific | 100      |
| 0xC8 to 0xFF                                                                | 200 to 255      | Reserved        | 56       |
| 0x100 to 0x2FF                                                              | 240 to 767      | Open            | 512      |
| 0x300 to 0x4FF                                                              | 768 to 1 279    | Vendor Specific | 512      |
| 0x500 to 0x8FF                                                              | 1 280 to 2 303  | Open            | 1 024    |
| 0x900 to 0xCFF                                                              | 2 304 to 3 327  | Vendor Specific | 1 024    |
| 0xD00 to 0xFFFF                                                             | 3 328 to 65 535 | Reserved        | 62 208   |
| NOTE Reserved ranges can be further defined by the ODVA, Inc. organization. |                 |                 |          |

#### 4.1.10.1.4 Connection Point ranges

The Connection Point values shall be as shown in Table 190. Connection Point = 0x0000 is reserved and shall not be used.

**Table 190 – Connection Point ranges**

| Range (hex)                | Range (decimal)            | Open objects      | Quantity      |
|----------------------------|----------------------------|-------------------|---------------|
| 0x0000                     | 0                          | Not used          | 1             |
| 0x0001 to 0x0063           | 1 to 99                    | Defined by object | 99            |
| 0x0064 to 0x00C7           | 100 to 199                 | Vendor Specific   | 100           |
| 0x00C8 to 0x00D1           | 200 to 209                 | Defined by object | 10            |
| 0x00D2 to 0x00EF           | 210 to 239                 | Reserved          | 30            |
| 0x00F0 to 0x00FF           | 240 to 255                 | Vendor Specific   | 16            |
| 0x0100 to 0x02FF           | 256 to 767                 | Defined by object | 512           |
| 0x0300 to 0x04FF           | 768 to 1 279               | Vendor Specific   | 512           |
| 0x0500 to 0x7FFF           | 1 280 to 32 767            | Defined by object | 31 488        |
| 0x8000 to 0xFFFF           | 32 768 to 65 535           | Vendor Specific   | 32 768        |
| 0x00010000 to 0x000FFFFFFF | 65 536 to 1 048 575        | Defined by object | 983 040       |
| 0x00100000 to 0xFFFFFFFF   | 1 048 576 to 4 294 967 295 | Reserved          | 4 293 918 720 |

NOTE Reserved ranges can be further defined by the ODVA, Inc. organization.

#### 4.1.10.1.5 Service code ranges

The **Service code** shall be a unique hexadecimal value assigned to each service. The Service Code values shall be as shown in Table 191. Service Code = 0x00 is reserved and shall not be used. The object-specific service codes shall be unique only within the class which define them.

**Table 191 – Service code ranges**

| Range (hex)  | Range (decimal) | Meaning                                                                                   | Quantity |
|--------------|-----------------|-------------------------------------------------------------------------------------------|----------|
| 0x00         | 00              | Reserved                                                                                  | 1        |
| 0x01 to 0x31 | 01 to 49        | Open. These are referred to as <i>Common Services</i> . They are defined in IEC 61158-5-2 | 49       |
| 0x32 to 0x4A | 50 to 74        | Vendor Specific                                                                           | 25       |
| 0x4B to 0x63 | 75 to 99        | Object Class Specific                                                                     | 25       |
| 0x64 to 0x7F | 100 to 127      | Reserved                                                                                  | 28       |
| 0x80 to 0xFF | 128 to 255      | Reserved for response messages                                                            | 128      |

NOTE Reserved range can be further defined by the ODVA, Inc. organization.

#### 4.1.10.2 Code definitions

##### 4.1.10.2.1 Object classes

Table 192 defines the class codes for the object classes. All other class codes listed within the "open" range and not listed in Table 192 shall be reserved. The requirements for these objects are defined in IEC 61158-4-2 or IEC 61158-5-2.

The fourth column of Table 192 (Associated Communication object (ACO)), identifies objects that could have to be included in the Associated Communication Objects instance attribute (11) of the Port Object (0xF4). There are also cases where these objects may have instance numbering restrictions (e.g. instance 1 not allowed), in particular when there is more than one port of same underlying logical link technology (e.g. TCP/IP Interface object).

If any of these objects (ACOs) is present in a device, see the Port Object (0xF4) specifications in IEC 61158-4-2 for more information about:

- when the Port object is required;
- if the Associated Communication Objects instance attribute (11) is required;
- if the object needs to be included in the Associated Communication Objects instance attribute;
- whether there are any instance ID numbering restrictions on the object.

**Table 192 – Class codes**

| Class code (hex) | Class code (decimal) | Object name                                  | Associated Communication object <sup>d</sup> |
|------------------|----------------------|----------------------------------------------|----------------------------------------------|
| 0x01             | 1                    | Identity object                              | No                                           |
| 0x02             | 2                    | Message Router object                        | No                                           |
| 0x03             | 3                    | DeviceNet object <sup>a</sup>                | Yes                                          |
| 0x04             | 4                    | Assembly object                              | No                                           |
| 0x05             | 5                    | Connection object                            | No                                           |
| 0x06             | 6                    | Connection Manager object                    | No                                           |
| 0x0F             | 15                   | Parameter object                             | No                                           |
| 0x2B             | 43                   | Acknowledge Handler object                   | No                                           |
| 0x39             | 57                   | Safety Supervisor object <sup>b</sup>        | No                                           |
| 0x3A             | 58                   | Safety Validator object <sup>b</sup>         | No                                           |
| 0x42             | 66                   | Motion Device Axis object <sup>c</sup>       | No                                           |
| 0x43             | 67                   | Time Sync object                             | No                                           |
| 0x47             | 71                   | DLR object <sup>a</sup>                      | Yes                                          |
| 0x48             | 72                   | QoS object <sup>a</sup>                      | Yes                                          |
| 0x4C             | 76                   | SERCOS III Link object <sup>b</sup>          | Yes                                          |
| 0x56             | 86                   | PRP/HSR Protocol object <sup>a</sup>         | Yes                                          |
| 0x57             | 87                   | PRP/HSR Nodes Table object <sup>a</sup>      | Yes                                          |
| 0xF0             | 240                  | ControlNet object <sup>a</sup>               | Yes                                          |
| 0xF1             | 241                  | Keeper object <sup>a</sup>                   | Yes                                          |
| 0xF2             | 242                  | Scheduling object <sup>a</sup>               | Yes                                          |
| 0xF3             | 243                  | Connection Configuration object <sup>a</sup> | No                                           |
| 0xF4             | 244                  | Port object <sup>a</sup>                     | No                                           |
| 0xF5             | 245                  | TCP/IP Interface object <sup>a</sup>         | Yes                                          |
| 0xF6             | 246                  | Ethernet Link object <sup>a</sup>            | Yes                                          |
| 0x109            | 265                  | LLDP Management object <sup>a</sup>          | No                                           |
| 0x10A            | 266                  | LLDP Data Table object <sup>a</sup>          | No                                           |

- <sup>a</sup> This object class is part of system management.
- <sup>b</sup> This object class is specified in IEC 61784-3-2.
- <sup>c</sup> This object class is specified in IEC 61800-7-202.
- <sup>d</sup> An Associated Communication object is defined in the Port object (instance attribute 11) and may have restrictions on the use of instance 1.

#### 4.1.10.2.2 Predefined class attributes

Nine predefined Class Attribute IDs shall be reserved, and these predefined/reserved class attributes shall have the definitions listed in Table 193. Because these attributes are reserved, class Attribute ID numbers 1 through 7, 200 and 201 shall always be reserved. Therefore, if a class attribute is added to an object class specification, it shall start with Attribute ID #8 and be different from 200 or 201.

**Table 193 – Reserved class attributes for all object class definitions**

| Attribute ID | Name                                  | Data type     | Semantics of values                                                                                                                                                          |
|--------------|---------------------------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1            | Revision                              | UINT          | The starting value assigned to this attribute is one (1). If updates that require an increase in this value are made, then the value of this attribute increases by one (1). |
| 2            | Max Instance                          | UINT          | This short version shall be used if no instance greater than 65 535 is supported.                                                                                            |
| 3            | Number of Instances                   | UINT          | This short version shall be used if no instance greater than 65 535 is supported.                                                                                            |
| 4            | Optional attribute list               | STRUCT of     |                                                                                                                                                                              |
|              | Number of attributes                  | UINT          |                                                                                                                                                                              |
|              | Optional attributes                   | ARRAY of UINT |                                                                                                                                                                              |
| 5            | Optional service list                 | STRUCT of     |                                                                                                                                                                              |
|              | Number services                       | UINT          |                                                                                                                                                                              |
|              | Optional services                     | ARRAY of UINT |                                                                                                                                                                              |
| 6            | Maximum ID Number Class Attributes    | UINT          |                                                                                                                                                                              |
| 7            | Maximum ID Number Instance Attributes | UINT          |                                                                                                                                                                              |
| 201          | Max Instance                          | UDINT         | This long version shall be used if instances greater than 65 535 are supported.                                                                                              |
| 202          | Number of Instances                   | UDINT         | This long version shall be used if instances greater than 65 535 are supported.                                                                                              |

#### 4.1.10.2.3 OM\_Service codes

Codes of the common services for the deterministic control network objects shall be as shown in Table 194.

NOTE Table 194 lists the codes and names of the common services while IEC 61158-5-2 provides a general description of each service.

**Table 194 – Common services list**

| Common service code | Common service name              |
|---------------------|----------------------------------|
| 0x00                | reserved                         |
| 0x01                | Get_Attributes_All               |
| 0x02                | Set_Attributes_All               |
| 0x03                | Get_Attribute_List               |
| 0x04                | Set_Attribute_List               |
| 0x05                | Reset                            |
| 0x06                | Start                            |
| 0x07                | Stop                             |
| 0x08                | Create                           |
| 0x09                | Delete                           |
| 0x0A                | Multiple_Service_Packet          |
| 0x0B to 0x0C        | reserved                         |
| 0x0D                | Apply_Attributes                 |
| 0x0E                | Get_Attribute_Single             |
| 0x0F                | reserved                         |
| 0x10                | Set_Attribute_Single             |
| 0x11                | Find_Next_Object_Instance        |
| 0x12 to 0x14        | reserved                         |
| 0x15                | Restore                          |
| 0x16                | Save                             |
| 0x17                | NOP (No operation)               |
| 0x18                | Get_Member                       |
| 0x19                | Set_Member                       |
| 0x1A                | Insert_Member                    |
| 0x1B                | Remove_Member                    |
| 0x1C                | Group_Sync                       |
| 0x1D                | Get_Connection_Point_Member_List |
| 0x1E to 0x31        | reserved                         |

Codes of the object specific services for the Identity object shall be as shown in Table 195.

**Table 195 – Identity object specific services list**

| Object specific service code | Service name |
|------------------------------|--------------|
| 0x4B                         | Flash_LEDs   |

Codes of the object specific services for the Message Router object shall be as shown in Table 196.

**Table 196 – Message Router object specific services list**

| Object specific service code | Service name          |
|------------------------------|-----------------------|
| 0x4B                         | Symbolic_Translation  |
| 0x4C                         | Send_Receive_Fragment |

Codes of the object specific services for the Acknowledge Handler object shall be as shown in Table 197.

**Table 197 – Acknowledge Handler object specific services list**

| Object specific service code | Service name        |
|------------------------------|---------------------|
| 0x4B                         | Add_AckData_Path    |
| 0x4C                         | Remove_AckData_Path |

Code of the object specific service for the Parameter object shall be as shown in Table 198.

**Table 198 – Parameter object specific services list**

| Object specific service code | Service name    |
|------------------------------|-----------------|
| 0x4B                         | Get_Enum_String |

#### 4.1.10.2.4 CM\_Service codes

Codes of the services specific to the Connection Manager shall be as shown in Table 199.

NOTE Table 199 lists the codes and names of the services while IEC 61158-5-2 provides a general description of each service.

**Table 199 – Services specific to Connection Manager**

| Service code | Service name           |
|--------------|------------------------|
| 0x4E         | Forward_Close          |
| 0x52         | Unconnected_Send       |
| 0x54         | Forward_Open           |
| 0x56         | Get_Connection_Data    |
| 0x57         | Search_Connection_Data |
| 0x5A         | Get_Connection_Owner   |
| 0x5B         | Large_Forward_Open     |

#### 4.1.10.2.5 CO\_Service codes

Codes of the services specific to the Connection object shall be as shown in Table 200.

NOTE Table 200 lists the codes and names of the services while IEC 61158-5-2 provides a general description of each service.

**Table 200 – Services specific to Connection object**

| Service code | Service name                 |
|--------------|------------------------------|
| 0x4B         | Connection_Bind              |
| 0x4C         | Producing_Application_Lookup |
| 0x4E         | SafetyClose                  |
| 0x54         | SafetyOpen                   |

#### 4.1.10.3 Device types

In order to allow interoperability and interchangeability, a device type shall be used to identify similar devices which

- exhibit the same behavior;
- produce and/or consume the same basic set of data;
- contain the same basic set of configurable attributes.

The formal definition of this information is known as a Device Profile: all devices with the same device type number shall meet the requirements specified in the Device Profile for that device type.

"Device Type" is a required instance attribute of the Identity object as described in IEC 61158-5-2.

Device types shall be either publicly defined or vendor specific. Table 201 specifies the numbering scheme to be used for device type numbering.

**Table 201 – Device type numbering**

| Range           | Type                        | Quantity |
|-----------------|-----------------------------|----------|
| 0x00 to 0x63    | Publicly Defined – Reserved | 100      |
| 0x64 to 0xC7    | Vendor Specific             | 100      |
| 0xC8 to 0xFF    | Reserved                    | 56       |
| 0x100 to 0x2FF  | Publicly Defined – Reserved | 512      |
| 0x300 to 0x4FF  | Vendor Specific             | 512      |
| 0x500 to 0xFFFF | Reserved                    | 64 256   |

All publicly defined device types shall have the same meaning for all implementers and are reserved.

The Generic Device type (type number = 0x00) is deprecated for new devices. It was previously used to define a device that does not fit into any of the defined device types.

NOTE The actual definition of publicly defined device types and corresponding Device Profiles is outside the scope of this document. They are defined by the ODVA, Inc. organization.

Vendor specific device types may be developed and vendors need not publish them. Each vendor shall maintain their own vendor-specific device types and corresponding number allocation.

#### 4.1.10.4 Implementation profile numbers

Implementation Profiles describe collections of information and functionality that the device supports, independent of its Device Type. Unlike Device Profiles, Implementation Profiles group subsets of Type 2 technologies and characteristics, providing a high-level description of the capabilities of a device.

The building blocks of an Implementation Profile are its features, which represent logical groupings of functions. A function can be comprised of one or more objects, attributes and services and/or other requirements.

Each Implementation Profile is defined by a structure with two members:

- Profile number – a unique identifier (UINT);
- Features supported – a bit string (DWORD) where the meaning of the individual bits is profile specific, with the exception of bit 31 which is reserved.

This information will be used in the Implementation Profiles instance attribute of the Identity object to report supported profiles (see IEC 61158-5-2).

Table 202 specifies the numbering scheme to be used for implementation profile numbering.

**Table 202 – Implementation profile numbering**

| Range          | Type                        | Quantity |
|----------------|-----------------------------|----------|
| 0x00           | Reserved                    | 1        |
| 0x01 - 0x63    | Publicly Defined - Reserved | 99       |
| 0x64- 0xC7     | Vendor Specific             | 100      |
| 0xC8 - 0xFF    | Reserved                    | 56       |
| 0x100 - 0x2FF  | Publicly Defined – Reserved | 512      |
| 0x300 - 0x4FF  | Vendor Specific             | 512      |
| 0x500 - 0xFFFF | Reserved                    | 64 256   |

NOTE The list of public implementation profiles is managed by the ODVA, Inc. organization. Definition of the implementation profiles and their feature values is outside the scope of this document.

#### 4.1.11 Error codes

##### 4.1.11.1 CM errors

The error codes are returned with the response to a Connection Manager Service Request which resulted in an error. These error codes shall be used to help diagnose the problem with a Service Request. The error code shall be split into an 8 bit general status and one or more 16 bit words of extended status. Unless specified otherwise, only the first word of extended status shall be required. Additional words of extended status may be used to specify additional device specific information. All devices which originate messages shall be able to handle multiple words of extended status.

Table 203 provides a summary of the available error codes.

Only the type of node indicated in the "Detected by" column shall return the general status/extended status code. The "Detected by" column values are Originator, Router and/or Target. When detected by Router or Target devices these error codes will be contained in the response packet. When detected by Originator these error codes may be reported back to the application via an object interface (e.g. Connection Configuration object) or other internal interface.

**Table 203 – Connection Manager service request error codes**

| General Status | Extended Status  | Detected by      | Explanation and description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|------------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x00           |                  |                  | Service completed successfully.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 0x01           | 0x0000 to 0x00FF |                  | Obsolete                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0x01           | 0x0100           | Router<br>Target | CONNECTION IN USE OR DUPLICATE FORWARD OPEN<br><br>This extended status code shall be returned when an originator is trying to make a connection to a target with which the originator could have already established a connection (Non-null/matching Forward_Open – see 4.1.5.3.2.2).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0x01           | 0x0101 to 0x0102 |                  | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0x01           | 0x0103           | Target           | TRANSPORT CLASS AND TRIGGER COMBINATION NOT SUPPORTED<br><br>A transport class and trigger combination has been specified which is not supported by the target application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 0x01           | 0x0104           |                  | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0x01           | 0x0105           |                  | Reserved for IEC 61784-3-2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 0x01           | 0x0106           | Target           | OWNERSHIP CONFLICT<br><br>The connection cannot be established since another connection has exclusively allocated some of the resources required for this connection. An example of this would be that only one exclusive owner connection can control an output point on an I/O Module. If a second exclusive owner connection (or redundant owner connection) is attempted, this error shall be returned.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 0x01           | 0x0107           | Target           | TARGET CONNECTION NOT FOUND<br><br>This extended status code shall be returned in response to the Forward_Close request, when the connection that is to be closed is not found at the target node. Routers shall not generate this extended status code. If the specified connection is not found at the router, the close request shall still be forwarded using the path specified in the Forward_Close request.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 0x01           | 0x0108           | Router<br>Target | INVALID NETWORK CONNECTION PARAMETER<br><br>This extended status code shall be returned as the result of specifying a connection type, connection priority, redundant owner or fixed/variable that is not supported by the device.<br><br>NOTE This extended status code is "deprecated". It is highly recommended that 0x011F, 0x0120, 0x0121, 0x0122, 0x0123, 0x0124, 0x0125 or 0x0132 be used instead.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 0x01           | 0x0109           | Router<br>Target | INVALID CONNECTION SIZE<br><br>This extended status code is returned when the target or router does not support the specified connection size. This could occur at a target because the size does not match the required size for a fixed size connection. It could occur at a router if the requested size is too large for the specified network.<br><br>An additional status word may follow indicating the maximum connection size supported by the responding node. The additional status word is required when issued in response to the Large_Forward_Open.<br><br>NOTE This extended status code is "deprecated". It is highly recommended that 0x0126, 0x0127, or 0x0128 be used instead. The only allowed exception is for Type 2/Type 15 translators. These translators cannot use 0x0126, 0x0127 or 0x0128 because expected sizes are unknown. |
| 0x01           | 0x010A to 0x010F |                  | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0x01           | 0x0110           | Target           | TARGET FOR CONNECTION NOT CONFIGURED<br><br>This extended status code shall be returned when a connection is requested to a target application that has not been configured and the connection request does not contain a data segment for configuration (see 4.1.9.7).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| General Status | Extended Status | Detected by                                                                                                                                                                 | Explanation and description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01           | 0x0111          | Router                                                                                                                                                                      | RPI NOT SUPPORTED.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                |                 | Target                                                                                                                                                                      | <p>This extended status code shall be returned if the device can not support the requested <math>O \Rightarrow T</math> or <math>T \Rightarrow O</math> RPI. This extended status code may also be used if the connection time-out multiplier produces a time-out value that is not supported by the device or the production inhibit time is not valid.</p> <p>It is highly recommended to use Extended Status 0x0112 when the RPI value(s) are not acceptable.</p> <p>Use of this extended status code when the connection time-out multiplier is not supported is deprecated. It is highly recommended that 0x0133 be used instead.</p> <p>Use of this extended status code when the production inhibit time is not valid is deprecated. It is highly recommended that 0x011B be used instead.</p> |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 0x01           | 0x0112          | Router<br><br>Target                                                                                                                                                        | RPI VALUE(S) NOT ACCEPTABLE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                |                 |                                                                                                                                                                             | <p>This extended status code shall be returned when the RPI value(s) in the Forward Open request are outside the range required by the application in the target device or the target is producing at a different interval. The target shall include information with acceptable RPI(s). For this error, the extended status size is 6 16-bit words and is formatted as follows.</p>                                                                                                                                                                                                                                                                                                                                                                                                                  |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                |                 |                                                                                                                                                                             | Data type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Value    | Explanation of field                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                |                 |                                                                                                                                                                             | UINT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 0x0112   | Extended status code                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                |                 |                                                                                                                                                                             | USINT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | variable | <p>Acceptable Originator to Target RPI (see below) type:</p> <p>0 – the RPI specified in the Forward Open was acceptable (the Originator to Target RPI value is ignored). <sup>a</sup></p> <p>1 – unspecified (used to suggest an alternate RPI, e.g. default)</p> <p>2 – minimum acceptable RPI (used when RPI was too fast for range)</p> <p>3 – maximum acceptable RPI (used when RPI was too slow for range)</p> <p>4 – required RPI to correct mismatch (used when data already being consumed at a different interval)</p> <p>5 to 255 – reserved</p>                      |
|                |                 |                                                                                                                                                                             | USINT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | variable | <p>Acceptable Target to Originator RPI (see below) type:</p> <p>0 – the RPI specified in the Forward Open was acceptable (the Target to Originator RPI value is ignored). <sup>a</sup></p> <p>1 – unspecified (used to suggest an alternate RPI, e.g. default)</p> <p>2 – minimum acceptable RPI (used when RPI was too fast for range)</p> <p>3 – maximum acceptable RPI (used when RPI was too slow for range)</p> <p>4 – required RPI to correct mismatch (used when data already being produced at a different interval, typically multicast)</p> <p>5 to 255 – reserved</p> |
|                |                 |                                                                                                                                                                             | UDINT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | variable | Value of Originator to Target RPI that is within the acceptable range for the application. This field is defined the same as the RPI parameter in the Forward Open request.                                                                                                                                                                                                                                                                                                                                                                                                      |
| UDINT          | variable        | Value of Target to Originator RPI that is within the acceptable range for the application. This field is defined the same as the RPI parameter in the Forward Open request. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

| General Status | Extended Status | Detected by                    | Explanation and description                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------|-----------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                |                 |                                | <sup>a</sup> The value of the Originator to Target and Target to Originator types shall not both be 0.                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0x01           | 0x0113          | Originator<br>Router<br>Target | OUT OF CONNECTIONS<br><br>Connection Manager cannot support any more connections. The maximum number of connections supported by the Connection Manager has already been created.                                                                                                                                                                                                                                                                                                                        |
| 0x01           | 0x0114          | Router<br>Target               | VENDOR ID OR PRODUCT CODE MISMATCH<br><br>The Product Code or Vendor ID specified in the electronic key logical segment does not match the Product Code or Vendor ID of the device. If the compatibility bit is set this extended status code is returned when the requested Vendor ID or Product Code is 0 or the device cannot emulate the specified Vendor ID or Product Code.                                                                                                                        |
| 0x01           | 0x0115          | Router<br>Target               | DEVICE TYPE MISMATCH<br><br>The Device Type specified in the electronic key logical segment does not match the Device Type of the device. If the compatibility bit is set this extended status code is returned when the requested Device Type is 0 or the device cannot emulate the specified Device Type.<br><br>EXCEPTION: If the compatibility bit is set, Device Type 0 is allowed if the recipient can emulate a device that reports the deprecated Generic Device Type Number.                    |
| 0x01           | 0x0116          | Router<br>Target               | REVISION MISMATCH<br><br>The major and minor revision specified in the electronic key logical segment does not correspond to a valid revision of the device. If the compatibility bit is set this extended status code is returned when the requested Major Revision and/or Minor Revision is 0 or the device cannot emulate the specified Major Revision.                                                                                                                                               |
| 0x01           | 0x0117          | Target                         | INVALID PRODUCED OR CONSUMED APPLICATION PATH<br><br>The produced or consumed application path specified in the connection path does not correspond to a valid produced or consumed application path within the target application. This error could also be returned if a produced or consumed application path was required, but not provided by a connection request.<br><br>NOTE This extended status code is "deprecated". It is highly recommended that 0x012A, 0x012B, or 0x012F be used instead. |
| 0x01           | 0x0118          | Target                         | INVALID OR INCONSISTENT CONFIGURATION APPLICATION PATH<br><br>An application path specified for the configuration data does not correspond to a configuration application or is inconsistent with the consumed or produced application paths. For example the connection path specifies float configuration data while the produced or consumed paths specify integer data.<br><br>NOTE This extended status code is "deprecated". It is highly recommended that 0x0129 or 0x012F be used instead.       |
| 0x01           | 0x0119          | Target                         | NON-LISTEN ONLY CONNECTION NOT OPENED<br><br>Connection request fails since there are no non-Listen Only connections with the same T⇒O application path currently open. See IEC 61158-5-2, 6.3.1.4.5, for a description of application connection types.<br><br>The extended status code shall be returned when an attempt is made to establish a Listen Only connection type to a target, which has no non-Listen Only connections with the same T⇒O application path already established.              |
| 0x01           | 0x011A          | Target                         | TARGET OBJECT OUT OF CONNECTIONS<br><br>The maximum number of connections supported by this instance of the target object has been exceeded.<br><br>For example, the Connection Manager could support 20 connections while the target object can only support 10 connections. On the 11 <sup>th</sup> Connection Request to the target object, this extended status code would be used to signify that the maximum number of connections already exist to the target object.                             |
| 0x01           | 0x011B          | Target                         | THE PRODUCTION INHIBIT TIME IS GREATER THAN THE RPI<br><br>The Production Inhibit Time is greater than the Target to Originator RPI.                                                                                                                                                                                                                                                                                                                                                                     |

| General Status | Extended Status | Detected by      | Explanation and description                                                                                                                                                                                                                                                                                                                                                 |              |                             |
|----------------|-----------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|-----------------------------|
| 0x01           | 0x011C          | Router<br>Target | TRANSPORT CLASS NOT SUPPORTED<br>The transport class requested in the Transport Type/Trigger parameter is not supported.                                                                                                                                                                                                                                                    |              |                             |
| 0x01           | 0x011D          | Router<br>Target | T⇒O PRODUCTION TRIGGER NOT SUPPORTED<br>The T⇒O production trigger requested in the Transport Type/Trigger parameter is not supported.                                                                                                                                                                                                                                      |              |                             |
| 0x01           | 0x011E          | Target           | DIRECTION NOT SUPPORTED<br>The direction requested in the Transport Type/Trigger parameter is not supported.                                                                                                                                                                                                                                                                |              |                             |
| 0x01           | 0x011F          | Target           | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION FIXVAR<br>This extended status code shall be returned as the result of specifying an O⇒T fixed / variable flag that is not supported.                                                                                                                                                                                       |              |                             |
| 0x01           | 0x0120          | Target           | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION FIXVAR<br>This extended status code shall be returned as the result of specifying a T⇒O fixed / variable flag that is not supported.                                                                                                                                                                                        |              |                             |
| 0x01           | 0x0121          | Target           | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION PRIORITY<br>This extended status code shall be returned as the result of specifying an O⇒T priority code that is not supported.                                                                                                                                                                                             |              |                             |
| 0x01           | 0x0122          | Target           | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION PRIORITY<br>This extended status code shall be returned as the result of specifying a T⇒O priority code that is not supported.                                                                                                                                                                                              |              |                             |
| 0x01           | 0x0123          | Target           | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION TYPE<br>This extended status code shall be returned as the result of specifying an O⇒T connection type that is not supported.                                                                                                                                                                                               |              |                             |
| 0x01           | 0x0124          | Router<br>Target | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION TYPE<br>This extended status code shall be returned as the result of specifying a T⇒O connection type that is not supported.                                                                                                                                                                                                |              |                             |
| 0x01           | 0x0125          | Router<br>Target | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION REDUNDANT_OWNER<br>This extended status code shall be returned as the result of specifying an O⇒T Redundant Owner flag that is not supported.                                                                                                                                                                               |              |                             |
| 0x01           | 0x0126          | Target           | INVALID CONFIGURATION SIZE<br>This extended status code is returned when the target device determines that the data segment provided in the Connection_Path parameter did not contain an acceptable number of 16-bit words for the configuration application path requested.<br>An additional status word shall follow indicating the maximum configuration size supported. |              |                             |
|                |                 |                  | <b>Data type</b>                                                                                                                                                                                                                                                                                                                                                            | <b>Value</b> | <b>Explanation of field</b> |
|                |                 |                  | UINT                                                                                                                                                                                                                                                                                                                                                                        | 0x0126       | Extended Status Code        |
|                |                 |                  | UINT                                                                                                                                                                                                                                                                                                                                                                        | Size         | Max size in words           |

| General Status | Extended Status | Detected by      | Explanation and description                                                                                                                                                                                                                                                                    |        |                      |
|----------------|-----------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|----------------------|
| 0x01           | 0x0127          | Router<br>Target | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION SIZE                                                                                                                                                                                                                                           |        |                      |
|                |                 |                  | This extended status code is returned by the target when the size of the consuming object declared in the Forward_Open request and available on the target does not match the size declared in the O⇒T Network Connection Parameter.                                                           |        |                      |
|                |                 |                  | This extended status code is returned by a router when it cannot support the size requested in the O⇒T Network Connection Parameter.                                                                                                                                                           |        |                      |
|                |                 |                  | An additional status word shall follow indicating the maximum originator to target size supported.                                                                                                                                                                                             |        |                      |
| 0x01           | 0x0128          | Router<br>Target | Data type                                                                                                                                                                                                                                                                                      | Value  | Explanation of field |
|                |                 |                  | UINT                                                                                                                                                                                                                                                                                           | 0x0127 | Extended Status Code |
|                |                 |                  | UINT                                                                                                                                                                                                                                                                                           | Size   | Max size in octets   |
|                |                 |                  |                                                                                                                                                                                                                                                                                                |        |                      |
| 0x01           | 0x0128          | Router<br>Target | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION SIZE                                                                                                                                                                                                                                           |        |                      |
|                |                 |                  | This extended status code is returned by the target when the size of the producing object declared in the Forward_Open request and available on the target does not match the size declared in the T⇒O Network Connection Parameter.                                                           |        |                      |
|                |                 |                  | This extended status code is returned by a router when it cannot support the size requested in the T⇒O Network Connection Parameter.                                                                                                                                                           |        |                      |
|                |                 |                  | An additional status word shall follow indicating the maximum target to originator size supported.                                                                                                                                                                                             |        |                      |
| 0x01           | 0x0128          | Router<br>Target | Data type                                                                                                                                                                                                                                                                                      | Value  | Explanation of field |
|                |                 |                  | UINT                                                                                                                                                                                                                                                                                           | 0x0128 | Extended Status Code |
|                |                 |                  | UINT                                                                                                                                                                                                                                                                                           | Size   | Max size in octets   |
|                |                 |                  |                                                                                                                                                                                                                                                                                                |        |                      |
| 0x01           | 0x0129          | Target           | INVALID CONFIGURATION APPLICATION PATH                                                                                                                                                                                                                                                         |        |                      |
|                |                 |                  | The configuration application path specified in the connection path does not correspond to a valid configuration application path within the target application. This error could also be returned if a configuration application path was required, but not provided by a connection request. |        |                      |
| 0x01           | 0x012A          | Target           | INVALID CONSUMING APPLICATION PATH                                                                                                                                                                                                                                                             |        |                      |
|                |                 |                  | The consumed application path specified in the connection path does not correspond to a valid consumed application path within the target application. This error could also be returned if a consumed application path was required, but not provided by a connection request.                |        |                      |
| 0x01           | 0x012B          | Target           | INVALID PRODUCING APPLICATION PATH                                                                                                                                                                                                                                                             |        |                      |
|                |                 |                  | The produced application path specified in the connection path does not correspond to a valid produced application path within the target application. This error could also be returned if a produced application path was required, but not provided by a connection request.                |        |                      |
| 0x01           | 0x012C          | Target           | CONFIGURATION SYMBOL DOES NOT EXIST                                                                                                                                                                                                                                                            |        |                      |
|                |                 |                  | Configuration Symbol does not exist. The originator attempts to connect to a configuration tag name, but the name is not on the list of tags defined on the target.                                                                                                                            |        |                      |
| 0x01           | 0x012D          | Target           | CONSUMING SYMBOL DOES NOT EXIST                                                                                                                                                                                                                                                                |        |                      |
|                |                 |                  | Consuming Symbol does not exist. The originator attempts to connect to a consuming tag name, but the name is not on the list of tags defined on the target.                                                                                                                                    |        |                      |
| 0x01           | 0x012E          | Target           | PRODUCING SYMBOL DOES NOT EXIST                                                                                                                                                                                                                                                                |        |                      |
|                |                 |                  | Producing Symbol does not exist. The originator attempts to connect to a producing tag name, but the name is not on the list of tags defined on the target.                                                                                                                                    |        |                      |
| 0x01           | 0x012F          | Target           | INCONSISTENT APPLICATION PATH COMBINATION                                                                                                                                                                                                                                                      |        |                      |
|                |                 |                  | The combination of configuration and/or consume and/or produce application paths specified in the connection path are inconsistent with each other.                                                                                                                                            |        |                      |

| General Status | Extended Status  | Detected by      | Explanation and description                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|------------------|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01           | 0x0130           | Target           | INCONSISTENT CONSUME DATA FORMAT<br>Information in the data segment is not consistent with the format of the consumed data. For example the configuration data specifies float configuration data while the consumed path specifies integer data.                                                                                                                                               |
| 0x01           | 0x0131           | Target           | INCONSISTENT PRODUCE DATA FORMAT<br>Information in the data segment is not consistent with the format of the produced data. For example the configuration data specifies float configuration data while the produced path specifies integer data.                                                                                                                                               |
| 0x01           | 0x0132           | Target           | NULL FORWARD OPEN FUNCTION NOT SUPPORTED<br>The target does not support the function requested by the Null Forward Open. The requested function may be "ping a device", "configure a device's application", or "reconfigure a target device's application".                                                                                                                                     |
| 0x01           | 0x0133           | Target<br>Router | CONNECTION TIMEOUT MULTIPLIER NOT ACCEPTABLE<br>This extended status code shall be returned as the result of specifying a connection timeout multiplier value that is reserved or that produces a timeout value that is too large to support in the device.                                                                                                                                     |
| 0x01           | 0x0134           | Target           | MISMATCHED T⇒O NETWORK CONNECTION SIZE<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T⇒O application path that is already being produced for another connection and the Size field in the T⇒O Network Connection Parameters in this request is valid but doesn't match the existing connection's Size field.                   |
| 0x01           | 0x0135           | Target           | MISMATCHED T⇒O NETWORK CONNECTION FIXVAR<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T⇒O application path that is already being produced for another connection and the Fixed/Variable bit in the T⇒O Network Connection Parameters in this request is valid but doesn't match the existing connection's Fixed/Variable bit. |
| 0x01           | 0x0136           | Target           | MISMATCHED T⇒O NETWORK CONNECTION PRIORITY<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T⇒O application path that is already being produced for another connection and the Priority field in the T⇒O Network Connection Parameters in this request is valid but doesn't match the existing connection's Priority field.       |
| 0x01           | 0x0137           | Target           | MISMATCHED TRANSPORT CLASS<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T⇒O application path that is already being produced for another connection and the Transport Class in this request is valid but doesn't match the existing connection's Transport Class.                                                              |
| 0x01           | 0x0138           | Target           | MISMATCHED T⇒O PRODUCTION TRIGGER<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T⇒O application path that is already being produced for another connection and the T⇒O Production Trigger in this request is valid but doesn't match the existing connection's T⇒O Production Trigger.                                         |
| 0x01           | 0x0139           | Target           | MISMATCHED T⇒O PRODUCTION INHIBIT TIME SEGMENT<br>This extended status code shall be returned when an originator requests to open a Multicast connection to an application path that is already being produced for another connection and the PIT in this request is valid but doesn't match the existing connection's PIT.                                                                     |
| 0x01           | 0x013A           | Router<br>Target | SERIAL NUMBER MISMATCH<br>The Serial Number specified in the electronic key logical segment does not match the Serial Number of the device.                                                                                                                                                                                                                                                     |
| 0x01           | 0x013B to 0x0202 |                  | Reserved                                                                                                                                                                                                                                                                                                                                                                                        |
| 0x01           | 0x0203           | Originator       | CONNECTION TIMED OUT<br>This extended status code shall occur when a connection has timed-out.                                                                                                                                                                                                                                                                                                  |

| General Status | Extended Status  | Detected by                    | Explanation and description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|------------------|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01           | 0x0204           | Originator<br>Router<br>Target | <p>UNCONNECTED REQUEST TIMED OUT</p> <p>An originator or router processing an Unconnected_Send, Forward_Open, Large_Forward_Open, or Forward_Close service shall return an Unconnected Request Timed Out error when the request cannot be delivered to the next device in the path (e.g the link specific retries fail), the allotted time is not sufficient to process the request, or no reply is received within the time specified by the Tick_time/Time-out_ticks parameters.</p> <p>A target processing a Forward_Open, Large_Forward_Open, or Forward_Close service may return an Unconnected Request Timed Out error when the allotted time is not sufficient to process the request.</p> <p>See 4.1.6.6 for more information.</p> |
| 0x01           | 0x0205           | Router                         | <p>PARAMETER ERROR IN UNCONNECTED REQUEST SERVICE</p> <p>For example, this shall be caused by a Connection Tick Time (see 4.1.6.6) and Connection time-out combination in an Unconnected_Send, Forward_Open, or Forward_Close service that is not supported by a router.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 0x01           | 0x0206           | Originator<br>Router           | <p>MESSAGE TOO LARGE FOR UNCONNECTED_SEND SERVICE</p> <p>This shall be caused when the Unconnected_Send is too large to be sent out on a network.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 0x01           | 0x0207           | Originator<br>Router           | <p>UNCONNECTED ACKNOWLEDGE WITHOUT RESPONSE</p> <p>The message was sent via the unconnected message service and an acknowledge was received but a data response message was not received.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 0x01           | 0x0208           | Target                         | <p>THIS SERVICE REQUIRES A CONNECTION</p> <p>The message was sent via an unconnected message service but the message shall be sent over a connection.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 0x01           | 0x0209 to 0x0300 |                                | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0x01           | 0x0301           | Originator<br>Router<br>Target | <p>NO BUFFER MEMORY AVAILABLE</p> <p>This extended status code shall occur when insufficient connection buffer memory is available in the device.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 0x01           | 0x0302           | Originator<br>Router<br>Target | <p>LINK TRANSMIT TIME NOT AVAILABLE FOR DATA</p> <p>This extended status code shall be returned by any device in the path that is a producer and can not allocate sufficient link transmit time for the connection on its link. This can only occur for connections that are specified as scheduled priority.</p>                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 0x01           | 0x0303           | Originator<br>Router<br>Target | <p>NO CONSUMED CONNECTION ID FILTER AVAILABLE</p> <p>Any device in the path that contains a link consumer for the connection and does not have an available consumed_connection_id filter available shall return this extended status code.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 0x01           | 0x0304           | Originator<br>Router<br>Target | <p>NOT CONFIGURED TO SEND SCHEDULED PRIORITY DATA</p> <p>If requested to make a connection that specifies scheduled priority, any device that is unable to send packets during the scheduled portion of the network update time interval shall return this extended status code.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 0x01           | 0x0305           | Router                         | <p>SCHEDULE SIGNATURE MISMATCH</p> <p>This extended status code shall be returned when the connection scheduling information in the originator device is not consistent with the connection scheduling information on the target network.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 0x01           | 0x0306           | Router                         | <p>SCHEDULE SIGNATURE VALIDATION NOT POSSIBLE</p> <p>This extended status code shall be returned when the connection scheduling information in the originator device can not be validated on the target network.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 0x01           | 0x0307           | Router                         | <p>STROBE OUTPUT NOT SUPPORTED</p> <p>Reserved for specific network communication profiles.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 0x01           | 0x0308 to 0x0309 |                                | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

| General Status | Extended Status | Detected by                    | Explanation and description                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------|-----------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01           | 0x0310          | Router<br>Target               | PARTICIPANT CONNECTION OPEN<br>At least one participant connection is open.                                                                                                                                                                                                                                                                                                                                                                  |
| 0x01           | 0x0311          | Originator<br>Router           | PORT NOT AVAILABLE<br>A Port specified in a Port Segment is Not Available or does not exist.                                                                                                                                                                                                                                                                                                                                                 |
| 0x01           | 0x0312          | Originator<br>Router           | LINK ADDRESS NOT VALID<br>Link Address specified in Port Segment Not Valid<br><br>This extended status code is the result of a port segment that specifies a link address that is not valid for the target network type. This extended status code shall not be used for link addresses that are valid for the target network type but do not respond.                                                                                       |
| 0x01           | 0x0313          | Router<br>Target               | DUPLICATE PARTICIPANT ID                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0x01           | 0x0314          | Router<br>Target               | PARTICIPANT ID NETWORK SEGMENT IS MISSING                                                                                                                                                                                                                                                                                                                                                                                                    |
| 0x01           | 0x0315          | Originator<br>Router<br>Target | INVALID SEGMENT IN CONNECTION PATH<br>Invalid Segment Type or Segment Value in Connection Path<br><br>This extended status code is the result of a device being unable to decode the connection path. For example, this could be caused by an unrecognized path type or a segment type occurring unexpectedly. This extended status code shall only be used when no other more specific extended status code provided in this table applies. |
| 0x01           | 0x0316          | Router<br>Target               | FORWARD CLOSE SERVICE CONNECTION PATH MISMATCH<br>The connection path in the Forward_Close service does not match the connection path in the connection being closed. This extended status error code has been "deprecated" because the Forward_Close service uses the connection triad for matching and doesn't use the connection path.                                                                                                    |
| 0x01           | 0x0317          | Originator<br>Router<br>Target | SCHEDULING NOT SPECIFIED<br>Either the Schedule Network Segment was not present or the Encoded Value in the Schedule Network Segment is invalid (i.e. 0).                                                                                                                                                                                                                                                                                    |
| 0x01           | 0x0318          | Originator<br>Router           | LINK ADDRESS TO SELF INVALID<br>Under some conditions (depends on the device), a link address in the Port Segment which points to the same device (loopback to oneself) is invalid.                                                                                                                                                                                                                                                          |
| 0x01           | 0x0319          | Router<br>Target               | SECONDARY RESOURCES UNAVAILABLE<br>In a dual chassis redundant system, a connection request that is made to the primary system shall be duplicated on the secondary system. If the secondary system is unable to duplicate the connection request, then this extended status code shall be returned.                                                                                                                                         |
| 0x01           | 0x031A          | Target                         | RACK CONNECTION ALREADY ESTABLISHED<br>A request for a module connection has been refused because part of the corresponding data is already included in a rack connection.                                                                                                                                                                                                                                                                   |
| 0x01           | 0x031B          | Target                         | MODULE CONNECTION ALREADY ESTABLISHED<br>A request for a rack connection has been refused because part of the corresponding data is already included in a module connection.                                                                                                                                                                                                                                                                 |
| 0x01           | 0x031C          | Originator<br>Router<br>Target | MISCELLANEOUS<br>This extended status is returned when no other extended status code applies for a connection related error.                                                                                                                                                                                                                                                                                                                 |

| General Status | Extended Status  | Detected by          | Explanation and description                                                                                                                                                                                                                                                                                               |
|----------------|------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01           | 0x031D           | Target               | REDUNDANT CONNECTION MISMATCH<br>This extended status code shall be returned when the following fields do not match when attempting to establish a redundant owner connection to the same target path:<br>O=>T_RPI;<br>O=>T_connection_parameters;<br>T=>O_RPI;<br>T=>O_connection_parameters;<br>xport_type_and_trigger. |
| 0x01           | 0x031E           | Target               | NO MORE USER CONFIGURABLE LINK CONSUMER RESOURCES AVAILABLE IN THE PRODUCING MODULE<br>A target shall return this extended status when the configured number of consumers for a producing application are already in use.                                                                                                 |
| 0x01           | 0x031F           | Target               | NO USER CONFIGURABLE LINK CONSUMER RESOURCES CONFIGURED IN THE PRODUCING MODULE<br>A target shall return this extended status when there are no consumers configured for a producing application to use.                                                                                                                  |
| 0x01           | 0x0320 to 0x07FF |                      | Vendor specific                                                                                                                                                                                                                                                                                                           |
| 0x01           | 0x0800           | Originator<br>Router | NETWORK LINK OFFLINE<br>Network link in path to module is offline                                                                                                                                                                                                                                                         |
| 0x01           | 0x0801 to 0x080F |                      | Reserved for IEC 61784-3-2                                                                                                                                                                                                                                                                                                |
| 0x01           | 0x0810           | Target               | NO TARGET APPLICATION DATA AVAILABLE<br>This extended status code is returned when the target application does not have valid data to produce for the requested connection.                                                                                                                                               |
| 0x01           | 0x0811           | Originator           | NO ORIGINATOR APPLICATION DATA AVAILABLE<br>This extended status code is returned when the originator application does not have valid data to produce for the requested connection.                                                                                                                                       |
| 0x01           | 0x0812           | Originator<br>Router | NODE ADDRESS HAS CHANGED SINCE THE NETWORK WAS SCHEDULED<br>A router on a scheduled network has a different node address than the value configured in the connection originator.                                                                                                                                          |
| 0x01           | 0x0813           | Router<br>Target     | NOT CONFIGURED FOR OFF-SUBNET MULTICAST<br>A multicast connection has been requested between a producer and a consumer that are on different subnets, and the producer is not configured for off-subnet multicast.                                                                                                        |
| 0x01           | 0x0814           | Target               | INVALID PRODUCE/CONSUME DATA FORMAT<br>Information in the data segment indicates that the format of the produced and/or consumed data is not valid.<br>NOTE This extended status code is "deprecated". It is highly recommended that 0x0130 or 0x0131 be used instead.                                                    |
| 0x01           | 0x0815 to 0x0834 |                      | Reserved for IEC 61784-3-2                                                                                                                                                                                                                                                                                                |
| 0x01           | 0x0835 to 0x08FF |                      | Reserved                                                                                                                                                                                                                                                                                                                  |
| 0x01           | 0x0900 to 0x09FF |                      | Reserved for security related status                                                                                                                                                                                                                                                                                      |
| 0x01           | 0x0A00           | Target               | UDP PORT FOR CLASS 0/1 NOT OPEN<br>The UDP port required to open the requested Class 0 or Class 1 connection has been disabled.                                                                                                                                                                                           |
| 0x01           | 0x0A01 to 0x0AFF |                      | Reserved for specific network communication profiles.                                                                                                                                                                                                                                                                     |

| General Status | Extended Status  | Detected by          | Explanation and description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01           | 0x0B00 to 0xFCFF |                      | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 0x01           | 0xFD00 to 0xFFFF |                      | Reserved (not to be used)                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0x02           | Empty            | Originator<br>Router | RESOURCE UNAVAILABLE FOR UNCONNECTED_SEND<br>The device lacks the resources to fully process the Unconnected Send Request.                                                                                                                                                                                                                                                                                                                                                    |
| 0x04           | Empty            | Router               | PATH SEGMENT ERROR IN UNCONNECTED SEND<br>Indicates the Type 2 router experienced a parsing error when extracting the Explicit Messaging Request from the Unconnected Send Request Service Data.                                                                                                                                                                                                                                                                              |
| 0x09           | Index to Element | Target               | ERROR IN DATA SEGMENT.<br>This general status code shall be returned when there is an error in the data segment of a forward open.<br>The Extended Status shall be the index to where the error was encountered in the Data Segment (see 4.1.9.7).                                                                                                                                                                                                                            |
| 0x0C           | Optional         | Target               | OBJECT STATE ERROR<br>This general status code shall be returned when the state of the target object of the connection prevents the service request from being handled. The Extended Status reports the object's present state. The extended status is optional.<br>For example, a target (application) object of the connection can need to be in an edit mode before attributes can be set. This is different from a service being rejected due to the state of the device. |
| 0x10           | Optional         | Router<br>Target     | DEVICE STATE ERROR<br>This general status code shall be returned when the state of the device prevents the service request from being handled. The Extended Status reports the device's present state. The extended status is optional.<br>For example, a controller may have a key switch which when set to the "hard run" state causes Service Requests to several different objects to fail (i.e. program edits). This general status code would then be returned.         |
| 0x13           | None             | Router<br>Target     | NOT ENOUGH DATA<br>The service did not supply enough data to perform the specified operation.                                                                                                                                                                                                                                                                                                                                                                                 |
| 0x15           | None             | Router<br>Target     | TOO MUCH DATA<br>The service supplied more data than was expected.                                                                                                                                                                                                                                                                                                                                                                                                            |

#### 4.1.11.2 OM errors

##### 4.1.11.2.1 General status format

General status used in the service primitives is specified in Table 204.

**Table 204 – General status codes**

| Status code | Name                                           | Description and meaning of status code                                                                                                                                                                                                                                                                            |
|-------------|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x00        | Success                                        | Service was successfully performed by the object specified                                                                                                                                                                                                                                                        |
| 0x01        | Communication related problem                  | A communications subsystem related problem was detected along the communications path.<br><br>This general status code is unique, as all extended status codes are scoped at the device level instead of the object level. They are all documented in Table 203 (Connection Manager service request error codes). |
| 0x02        | Resource unavailable                           | Resources needed for the object to perform the requested service were unavailable                                                                                                                                                                                                                                 |
| 0x03        | Invalid parameter value                        | See status code 0x20, which is the preferred value to use for this condition                                                                                                                                                                                                                                      |
| 0x04        | Path segment error                             | The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered                                                                                                                                                  |
| 0x05        | Path destination unknown                       | The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered                                                                                              |
| 0x06        | Partial transfer                               | Only part of the expected data was transferred                                                                                                                                                                                                                                                                    |
| 0x07        | Connection lost                                | The messaging connection was lost                                                                                                                                                                                                                                                                                 |
| 0x08        | Service not supported                          | The requested service was not implemented or was not defined for this class or object instance                                                                                                                                                                                                                    |
| 0x09        | Invalid attribute value                        | Invalid attribute data detected                                                                                                                                                                                                                                                                                   |
| 0x0A        | Attribute list error                           | An attribute in the Get_Attribute_List or Set_Attribute_List response has a non zero status                                                                                                                                                                                                                       |
| 0x0B        | Already in requested mode/state                | The object is already in the mode/state being requested by the service                                                                                                                                                                                                                                            |
| 0x0C        | Object state conflict                          | The object cannot perform the requested service in its current mode/state                                                                                                                                                                                                                                         |
| 0x0D        | Object already exists                          | The requested instance of object to be created already exists                                                                                                                                                                                                                                                     |
| 0x0E        | Attribute not settable                         | A request to modify a non-modifiable attribute was received                                                                                                                                                                                                                                                       |
| 0x0F        | Privilege violation                            | A permission/privilege check failed                                                                                                                                                                                                                                                                               |
| 0x10        | Device state conflict                          | The device's current mode/state prohibits the execution of the requested service                                                                                                                                                                                                                                  |
| 0x11        | Response data too large                        | The data to be transmitted in the response buffer is larger than the allocated response buffer                                                                                                                                                                                                                    |
| 0x12        | Fragmentation of a primitive value             | The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type                                                                                                                                                                                                   |
| 0x13        | Not enough data                                | The service did not supply enough data to perform the specified operation                                                                                                                                                                                                                                         |
| 0x14        | Attribute not supported                        | The attribute specified in the request is not supported                                                                                                                                                                                                                                                           |
| 0x15        | Too much data                                  | The service supplied more data than was expected                                                                                                                                                                                                                                                                  |
| 0x16        | Object instance does not exist                 | The object instance specified does not exist in the device. This error code shall be used instead of 0x05 (Path Destination Unknown) when the class is supported but the specific instance does not exist.                                                                                                        |
| 0x17        | Service fragmentation sequence not in progress | The fragmentation sequence for this service is not currently active for this data                                                                                                                                                                                                                                 |
| 0x18        | No stored attribute data                       | The attribute data of this object was not saved prior to the requested service                                                                                                                                                                                                                                    |
| 0x19        | Store operation failure                        | The attribute data of this object was not saved due to a failure during the attempt.                                                                                                                                                                                                                              |

| Status code  | Name                                       | Description and meaning of status code                                                                                                                                                                                                                                                                                        |
|--------------|--------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x1A         | Routing failure, request packet too large  | The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service                                                                                                                                                                     |
| 0x1B         | Routing failure, response packet too large | The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service                                                                                                                                                                  |
| 0x1C         | Missing attribute list entry data          | The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior                                                                                                                                                                                              |
| 0x1D         | Invalid attribute value list               | The service is returning the list of attributes supplied with status information for those attributes that were invalid                                                                                                                                                                                                       |
| 0x1E         | Embedded service error                     | An embedded service resulted in an error                                                                                                                                                                                                                                                                                      |
| 0x1F         | Vendor specific error                      | A vendor specific error has been encountered. The extended status code field of the error response defines the particular error encountered. Use of this general status code should only be performed when none of the status codes presented in this table or within an object class definition accurately reflect the error |
| 0x20         | Invalid parameter                          | A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an application object specification                                                                                                           |
| 0x21         | Write once value or medium already written | An attempt was made to write to a write once medium (e.g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established                                                                                                                                                       |
| 0x22         | Invalid Response Received                  | An invalid response is received (e.g. response service code does not match the request service code, or response message is shorter than the minimum expected response size). This status code can serve for other causes of invalid responses                                                                                |
| 0x23         | Buffer overflow                            | The message received is larger than the receiving buffer can handle. The entire message was discarded.                                                                                                                                                                                                                        |
| 0x24         | Message format error                       | The format of the received message is not supported by the server.                                                                                                                                                                                                                                                            |
| 0x25         | Key Failure in path                        | The Key Segment which was included as the first segment in the path does not match the destination module. The extended status shall indicate which part of the key check failed (see Table 205).                                                                                                                             |
| 0x26         | Path Size Invalid                          | The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.                                                                                                                                                 |
| 0x27         | Unexpected attribute in list               | An attempt was made to set an attribute that cannot be set at this time.                                                                                                                                                                                                                                                      |
| 0x28         | Invalid Member ID                          | The Member ID specified in the request does not exist in the specified Class/Instance/Attribute.                                                                                                                                                                                                                              |
| 0x29         | Member not settable                        | A request to modify a non-modifiable member was received.                                                                                                                                                                                                                                                                     |
| 0x2A         | Group 2 only server general failure        | Reserved for specific network communication profiles.                                                                                                                                                                                                                                                                         |
| 0x2B         | Unknown Type 15 error                      | A Type 2 to Type 15 translator received an unknown Type 15 Exception Code                                                                                                                                                                                                                                                     |
| 0x2C         | Attribute not gettable                     | A request to read a non-readable attribute was received                                                                                                                                                                                                                                                                       |
| 0x2D         | Instance not deletable                     | The requested object instance cannot be deleted                                                                                                                                                                                                                                                                               |
| 0x2E         | Service not supported for specified path   | The object supports the service, but not for the designated application path (e.g. attribute).<br><br>This code shall not be used when a more specific General Status code applies, e.g. 0x0E (Attribute not settable) or 0x29 (Member not settable).                                                                         |
| 0x2F to 0xCF | Reserved                                   | Reserved for future extensions                                                                                                                                                                                                                                                                                                |

| Status code  | Name                                         | Description and meaning of status code                                                                                                                                                                                                |
|--------------|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0xD0 to 0xFF | Reserved for object class and service errors | This range of status codes shall be used to indicate object class specific errors. Use of this range should only be performed when none of the status codes presented in this table accurately reflect the error that was encountered |

NOTE IEC 61158-5-2 contains more detail on the service response general status codes for each common service.

#### 4.1.11.2.2 Extended status format

The `MR_response_Header` contains a parameter called `Extended_status[]` which is labeled extended status data.

Actual usage and definition of this extended status is dependant on the object class or service: each object class defines its own extended status values and value ranges (including the vendor specific ones), except for extended status "0x00FF" which shall not be used by any object due to its special meaning for some network communication profiles. For a given object class, extended status are unique to each general status code, with the exception of general status code 0x01. All extended status values are reserved unless otherwise indicated within the object definition.

Table 205 specifies the format of the extended status for a general status of "Key Failure in path" (0x25).

**Table 205 – Extended status code for a general status of "Key Failure in path"**

| General status code | Extended status code | Status name                        | Explanation and description                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------|----------------------|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x25                | 0x0114               | VENDOR ID OR PRODUCT CODE MISMATCH | The Product Code or Vendor ID specified in the electronic key logical segment does not match the Product Code or Vendor ID of the target device. If the compatibility bit is set this extended status code is returned when the requested Vendor ID or Product Code is 0 or the device cannot emulate the specified Vendor ID or Product Code.                                                                                                                   |
| 0x25                | 0x0115               | DEVICE TYPE MISMATCH               | The Device Type specified in the electronic key logical segment does not match the Device Type of the target device. If the compatibility bit is set this extended status code is returned when the requested Device Type is 0 or the device cannot emulate the specified Device Type.<br><br>EXCEPTION: If the compatibility bit is set, Device Type 0 is allowed if the recipient can emulate a device that reports the deprecated Generic Device Type Number. |
| 0x25                | 0x0116               | REVISION MISMATCH                  | The major and minor revision specified in the electronic key logical segment does not correspond to a valid revision of the target device. If the compatibility bit is set this extended status code is returned when the requested Major revision and/or Minor revision is 0 or the device cannot emulate the specified revision.                                                                                                                               |
| 0x25                | 0x013A               | SERIAL NUMBER MISMATCH             | The Serial Number specified in the electronic key logical segment does not match the Serial Number of the device.                                                                                                                                                                                                                                                                                                                                                |

NOTE These extended status codes are the same defined for the Connection Manager general status code 0x01.

#### 4.1.11.2.3 Object-specific general and extended status format

##### 4.1.11.2.3.1 General

Subclause 4.1.11.2.3 specifies the format of object-specific general and extended status.

#### 4.1.11.2.3.2 Identity object status format

Table 206 specifies the format of object-specific general and extended status for the Identity object.

**Table 206 – Identity object status codes**

| General status code | 8-bit associated extended status codes | Status Name          | Description of Status                                                                             |
|---------------------|----------------------------------------|----------------------|---------------------------------------------------------------------------------------------------|
| 0x00 to 0xCF        |                                        | General status codes | Defined in 4.1.11.2.1                                                                             |
|                     | 0x00 to 0xEE                           |                      | Reserved extended status codes                                                                    |
|                     | 0xF0 to 0xFE                           | Vendor specific      | Vendor specific extended status codes                                                             |
|                     | 0xFF                                   |                      | Used with all general status codes when required and no other extended status code is assigned    |
| 0xD0                |                                        | Hardware diagnostic  | Device self testing and hardware diagnostic conditions                                            |
|                     | 0x00                                   |                      | Reserved                                                                                          |
|                     | 0x01                                   |                      | Checksum (or CRC) error – Code space/ROM – Boot section                                           |
|                     | 0x02                                   |                      | Checksum (or CRC) error – Code space/ROM – Application section                                    |
|                     | 0x03                                   |                      | Checksum (or CRC) error – NV (flash/EEPROM) memory                                                |
|                     | 0x04                                   |                      | Invalid non-volatile (NV) memory – Configuration bad                                              |
|                     | 0x05                                   |                      | Invalid non-volatile (NV) memory – No configuration established                                   |
|                     | 0x06                                   |                      | RAM memory bad – The RAM memory in the device was determined to be experiencing inoperative cells |
|                     | 0x07                                   |                      | ROM/Flash memory bad                                                                              |
|                     | 0x08                                   |                      | Flash/EEPROM (NV) Memory Bad                                                                      |
|                     | 0x09                                   |                      | Interconnect wiring error / signal path problem                                                   |
|                     | 0x0A                                   |                      | Power problem – Over current                                                                      |
|                     | 0x0B                                   |                      | Power problem – Over voltage                                                                      |
|                     | 0x0C                                   |                      | Power problem – Under voltage                                                                     |
|                     | 0x0D                                   |                      | Internal sensor problem                                                                           |
|                     | 0x0E                                   |                      | System clock fault                                                                                |
|                     | 0x0F                                   |                      | Hardware configuration does not match NV configuration                                            |
|                     | 0x10                                   |                      | Watchdog disabled/idle                                                                            |
|                     | 0x11                                   |                      | Watchdog timer expired                                                                            |
|                     | 0x12                                   |                      | Device over temperature                                                                           |
|                     | 0x13                                   |                      | Ambient temperature outside of operating limits                                                   |
|                     | 0x14 to 0xEF                           | (reserved)           | Reserved                                                                                          |
|                     | 0xF0 to 0xFE                           |                      | Vendor specific extended status codes                                                             |
|                     | 0xFF                                   |                      | Used with all general status codes when required and no other extended status code is assigned    |
| 0xD1                |                                        | Device status/states | Device status events and conditions                                                               |
|                     | 0x01                                   |                      | Power applied                                                                                     |
|                     | 0x02                                   |                      | Device reset                                                                                      |

| General status code | 8-bit associated extended status codes | Status Name                           | Description of Status                                                                                                                                                                                                                                                                                                         |
|---------------------|----------------------------------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | 0x03                                   |                                       | Device power loss                                                                                                                                                                                                                                                                                                             |
|                     | 0x04                                   |                                       | Activated                                                                                                                                                                                                                                                                                                                     |
|                     | 0x05                                   |                                       | Deactivated                                                                                                                                                                                                                                                                                                                   |
|                     | 0x06                                   |                                       | Enter self-test state                                                                                                                                                                                                                                                                                                         |
|                     | 0x07                                   |                                       | Enter standby state                                                                                                                                                                                                                                                                                                           |
|                     | 0x08                                   |                                       | Enter operational state                                                                                                                                                                                                                                                                                                       |
|                     | 0x09                                   |                                       | Non-specific minor recoverable fault detected                                                                                                                                                                                                                                                                                 |
|                     | 0x0A                                   |                                       | Non-specific minor unrecoverable fault detected                                                                                                                                                                                                                                                                               |
|                     | 0x0B                                   |                                       | Non-specific major recoverable fault detected                                                                                                                                                                                                                                                                                 |
|                     | 0x0C                                   |                                       | Non-specific major unrecoverable fault detected                                                                                                                                                                                                                                                                               |
|                     | 0x0D                                   |                                       | Fault(s) corrected                                                                                                                                                                                                                                                                                                            |
|                     | 0x0E                                   |                                       | CCV changed                                                                                                                                                                                                                                                                                                                   |
|                     | 0x0F                                   |                                       | Heartbeat interval changed                                                                                                                                                                                                                                                                                                    |
|                     | 0x10 to 0xEF                           | (reserved)                            |                                                                                                                                                                                                                                                                                                                               |
|                     | 0xF0 to 0xFE                           | Vendor specific                       | Vendor specific                                                                                                                                                                                                                                                                                                               |
|                     | 0xFF                                   |                                       | Used with all general status codes when required and no other extended status code is assigned                                                                                                                                                                                                                                |
| 0xD2 to 0xEF        |                                        | Object specific general status codes  | Reserved – Not yet assigned                                                                                                                                                                                                                                                                                                   |
|                     | 0x00 to 0xFF                           | Reserved                              |                                                                                                                                                                                                                                                                                                                               |
| 0xF0 to 0xFF        |                                        | Vendor specific general status codes  | A vendor specific error has been encountered. The extended status code field of the error response defines the particular error encountered. Use of this general status code should only be performed when none of the status codes presented in this table or within an object class definition accurately reflect the error |
|                     | 0x00 to 0xFF                           | Vendor specific extended status codes | All extended status codes are available for association with each vendor specific general status code                                                                                                                                                                                                                         |

## 4.2 Data abstract syntax specification

### 4.2.1 Transport format specification

The lower layers of open system architectures are concerned with the transport of user data among distributed functional units. In these layers, the user data can be regarded simply as a sequence of octets. However, application layer entities may manipulate the values of quite complex data types. To achieve independence between the application layer and lower layers, data types may be specified in an abstract syntax notation.

Supplementing the abstract syntax with one or more algorithms (called encoding rules) can determine the values of the lower layer octets which carry the application layer values. The combination of the abstract syntax with a single set of transfer rules produces a specific transfer syntax.

#### 4.2.2 Abstract syntax notation

The data type definitions provided in this document shall be written in Abstract Syntax Notation One (ASN.1), as defined in ISO/IEC 8824-1. These type definitions shall be a part of the ASN.1 module "Network DataTypes." The beginning ASN.1 statement indicating that these definitions are in this module is:

```
control network DataTypes DEFINITIONS ::= BEGIN
```

and the closing ASN.1 statement shall be the keyword "END".

The abstract definitions that follow shall comprise the set of control network data types. In addition, provision is made to extend or derive new data types based on existing defined types, and to include those in a "type dictionary."

#### 4.2.3 Control network data specification

The notation [typeId] for directly derived, enumerated, subrange and structured bit string data shall mean that the tag shall be taken from the "type" field in the corresponding VariableDictionaryEntry.

```
Network Data ::= CHOICE {ElementaryData, DerivedData}
ElementaryData ::= CHOICE {
    BOOL,
    FixedLengthInteger,
    FixedLengthReal,
    AnyTime,
    AnyDate,
    AnyString,
    FixedLengthBitString,
    EPATH}
DerivedData ::= CHOICE {
    DirectlyDerivedData,
    EnumeratedData,
    SubrangeData,
    StructuredBitStringData,
    ARRAY,
    STRUCT,
    FunctionBlockData}
DirectlyDerivedData ::= [typeId] NetworkData
EnumeratedData ::= [typeId] USINT
SubrangeData ::= [typeId] FixedLengthInteger
StructuredBitStringData ::= [typeId] FixedLengthBitString
FixedLengthInteger ::= CHOICE {SignedInteger, UnsignedInteger}
SignedInteger ::= CHOICE {SINT, INT, DINT, LINT}
UnsignedInteger ::= CHOICE {USINT, UINT, UDINT, ULINT}
FixedLengthReal ::= CHOICE {REAL, LREAL}
AnyTime ::= CHOICE {ITIME, TIME, FTIME, LTIME, NTIME, STIME, UTIME}
AnyDate ::= CHOICE {DATE, TIME_OF_DAY, DATE_AND_TIME}
AnyString ::= CHOICE {STRING, STRING2}
FixedLengthBitString ::= CHOICE {SWORD, WORD, DWORD, LWORD}
BOOL ::= [PRIVATE 1] IMPLICIT BOOLEAN
SINT ::= [PRIVATE 2] IMPLICIT OCTET STRING-- 1 octet
INT ::= [PRIVATE 3] IMPLICIT OCTET STRING-- 2 octets
```

DINT ::= [PRIVATE 4] IMPLICIT OCTET STRING -- 4 octets  
 LINT ::= [PRIVATE 5] IMPLICIT OCTET STRING -- 8 octets  
 USINT ::= [PRIVATE 6] IMPLICIT OCTET STRING -- 1 octet  
 UINT ::= [PRIVATE 7] IMPLICIT OCTET STRING -- 2 octets  
 UDINT ::= [PRIVATE 8] IMPLICIT OCTET STRING -- 4 octets  
 ULINT ::= [PRIVATE 9] IMPLICIT OCTET STRING -- 8 octets  
 REAL ::= [PRIVATE 10] IMPLICIT OCTET STRING -- 4 octets  
 LREAL ::= [PRIVATE 11] IMPLICIT OCTET STRING -- 8 octets  
 STIME ::= [PRIVATE 12] IMPLICIT ULINT  
 DATE ::= [PRIVATE 13] IMPLICIT UINT  
 TIME\_OF\_DAY ::= [PRIVATE 14] IMPLICIT UDINT  
 DATE\_AND\_TIME ::= [PRIVATE 15] IMPLICIT SEQUENCE {  
     time\_of\_day UDINT,  
     date UINT }  
 STRING ::= [PRIVATE 16] IMPLICIT SEQUENCE {  
     charcount UINT,  
     stringcontents OCTET STRING} -- one octet per character  
 SWORD ::= [PRIVATE 17] IMPLICIT OCTET STRING -- 1 octet  
 WORD ::= [PRIVATE 18] IMPLICIT OCTET STRING -- 2 octets  
 DWORD ::= [PRIVATE 19] IMPLICIT OCTET STRING -- 4 octets  
 LWORD ::= [PRIVATE 20] IMPLICIT OCTET STRING -- 8 octets  
 STRING2 ::= [PRIVATE 21] IMPLICIT SEQUENCE {  
     charcount UINT,  
     string2contents OCTET STRING} -- 2 octets/ character  
 FTIME ::= [PRIVATE 22] IMPLICIT DINT  
 LTIME ::= [PRIVATE 23] IMPLICIT LINT  
 ITIME ::= [PRIVATE 24] IMPLICIT INT  
 STRINGN ::= [PRIVATE 25] IMPLICIT SEQUENCE {  
     charsize UINT,  
     charcount UINT,  
     stringNcontents OCTET STRING} -- N octets/ character  
 SHORT\_STRING ::= [PRIVATE 26] IMPLICIT SEQUENCE {  
     charcount USINT,  
     stringcontents OCTET STRING} -- one octet per character  
 TIME ::= [PRIVATE 27] IMPLICIT DINT  
 EPATH ::= [PRIVATE 28] IMPLICIT OCTET STRING -- IEC 61158-6-2  
 STRINGI ::= [PRIVATE 30] IMPLICIT SEQUENCE {  
     Stringnum USINT (number of strings)  
     array of STRUCT  
     language1 USINT (first character from ISO 639-2/T)  
     language2 USINT (second character from ISO 639-2/T)  
     language3 USINT (third character from ISO 639-2/T)  
     datatype EPATH limited to the values 0xD0, 0xD5, 0xD9, and  
     0xDA)  
     charset UINT from IANA MIB Printer Codes  
     (IETF RFC 1759))  
     stringcontents CHOICE OF (SHORT\_STRING, STRING, STRING2,  
         or STRINGN) -- based on datatype field  
 NTIME ::= [PRIVATE 31] IMPLICIT LINT

```

UTIME ::= [PRIVATE 12] IMPLICIT ULINT
ARRAY ::= SEQUENCE OF NetworkData -- All of same base type
STRUCT ::= SEQUENCE OF NetworkData -- May be different types
FunctionBlockData ::= SET{
    inputs    [0] IMPLICIT STRUCT OPTIONAL,
    outputs    [1] IMPLICIT STRUCT OPTIONAL,
    controlInputs [2] IMPLICIT STRUCT OPTIONAL,
    controlOutputs [3] IMPLICIT STRUCT OPTIONAL}

```

#### 4.2.4 Data type specification / dictionaries

The definition of an object may include text that defines attributes. Attributes shall be assigned a **Data Type** in an object specification. The Data Type may be one of those defined in this document or may be an object specific extension to this document. The following definition shall provide a **Type Specification** for data and shall provide a structure for extending or deriving new data types based on existing defined types.

```

Dictionary ::= CHOICE {VariableDictionary, TypeDictionary}
VariableDictionary ::= SEQUENCE OF VariableDictionaryEntry
VariableDictionaryEntry ::= SEQUENCE{
    name AnyString,
    id FixedLengthInteger,
    type TypeID,
    ranges SEQUENCE OF Subrange, -- for arrays
    accessPrivilege BOOL {READ_ONLY(0), READ_WRITE(1)}
TypeID ::= OCTET STRING -- ASN.1 encoded tag value of the
-- DataTypeSpecification module
Subrange ::= SEQUENCE {
    minValue FixedLengthInteger,
    maxValue FixedLengthInteger}
TypeDictionary ::= SEQUENCE OF TypeDictionaryEntry
TypeDictionaryEntry ::= SEQUENCE {
    name AnyString,
    type TypeID,
    spec DataTypeSpecification}
DataTypeSpecification ::= CHOICE {
    alt [PRIVATE 0] IMPLICIT AlternateTypeSpec,
    bool [PRIVATE 1] IMPLICIT NULL, -- BOOL
    sint [PRIVATE 2] IMPLICIT NULL, -- SINT
    int [PRIVATE 3] IMPLICIT NULL, -- INT
    dint [PRIVATE 4] IMPLICIT NULL, -- DINT
    lint [PRIVATE 5] IMPLICIT NULL, -- LINT
    uint [PRIVATE 6] IMPLICIT NULL, -- USINT
    udint [PRIVATE 7] IMPLICIT NULL, -- UUINT
    ulint [PRIVATE 9] IMPLICIT NULL, -- ULINT
    real [PRIVATE 10] IMPLICIT NULL, -- REAL
    lreal [PRIVATE 11] IMPLICIT NULL, -- LREAL
    stime [PRIVATE 12] IMPLICIT NULL, -- STIME
    date [PRIVATE 13] IMPLICIT NULL, -- DATE
    tod [PRIVATE 14] IMPLICIT NULL, -- TIME_OF_DAY
    dat [PRIVATE 15] IMPLICIT NULL, -- DATE_AND_TIME
    str1 [PRIVATE 16] IMPLICIT NULL, -- STRING
    sword [PRIVATE 17] IMPLICIT NULL, -- SWORD
    word [PRIVATE 18] IMPLICIT NULL, -- WORD
    dword [PRIVATE 19] IMPLICIT NULL, -- DWORD
    lword [PRIVATE 20] IMPLICIT NULL, -- LWORD
    str2 [PRIVATE 21] IMPLICIT NULL, -- STRING2

```

```

ftime [PRIVATE 22]    IMPLICIT NULL, --    FTIME
ltime [PRIVATE 23]    IMPLICIT NULL, --    LTIME
itime [PRIVATE 24]    IMPLICIT NULL, --    ITIME
strN  [PRIVATE 25]    IMPLICIT NULL, --    STRINGN
shstr [PRIVATE 26]    IMPLICIT NULL, --    SHORT_STRING
time  [PRIVATE 27]    IMPLICIT NULL, --    TIME
epath [PRIVATE 28]    IMPLICIT NULL, --    EPATH
strI  [PRIVATE 30]    IMPLICIT NULL, --    STRINGI
ntime [PRIVATE 31]    IMPLICIT NULL, --    NTIME
utime [PRIVATE 32]    IMPLICIT NULL, --    UTIME
constructedData CHOICE {
    abbrevStruc [0]    IMPLICIT AbbreviatedStructTypeSpec,
    abbrevArr  [1]    IMPLICIT AbbreviatedArrayTypeSpec,
    frmlStruc  [2]    IMPLICIT FormalStructTypeSpec,
    frmlArr    [3]    IMPLICIT FormalArrayTypeSpec,
    expBitStr  [4]    IMPLICIT ExpandedFixedLenBitStrTypeSpec,
    expStr1    [5]    IMPLICIT ExpandedStringTypeSpec,
    expStr2    [6]    IMPLICIT ExpandedString2TypeSpec
    frmlHandleStruc [7] IMPLICIT FormalHandleStructTypeSpec}
}

```

AbbreviatedStructTypeSpec ::= UINT

AbbreviatedArrayTypeSpec ::= DataTypeSpecification

FormalStructTypeSpec ::= SEQUENCE OF DataTypeSpecification

FormalHandleStructTypeSpec ::= SEQUENCE OF DataTypeHandleSpecification

DataTypeHandleSpecification ::= DataTypeSpecification MemberHandle

MemberHandle ::= USINT, UINT, UDINT

FormalArrayTypeSpec ::= SEQUENCE {numberOfElements IMPLICIT

FixedLengthInteger, -- Array Number of Elements  
                   dataType     DataTypeSpecification }

ExpandedFixedLenBitStrTypeSpec ::= SEQUENCE {  
           bitStrType DataTypeSpecification     -- SWORD, WORD, DWORD, or  
 LWORD  
           bitFields.[7] IMPLICIT BitFieldDef}

BitFieldDef ::= SEQUENCE OF {  
           bitDef [2] IMPLICIT OCTET STRING}     -- Length is always 2  
 octets.

                  -- First octet contains starting  
                   -- Bit Position. Trailing octet  
                   -- contains the number of bits.

ExpandedStringTypeSpec ::= UINT-- String Length In Octets

ExpandedString2TypeSpec ::= UINT-- String Length In Octets

AlternateTypeSpec ::= CHOICE {  
           directlyDerivedTypeSpec [0]     IMPLICIT     TypeID,  
           subrangeTypeSpec        [1]     IMPLICIT     SubrangeTypeSpec ,  
           enumeratedTypeSpec       [2]     IMPLICIT     EnumeratedTypeSpec,  
           fbTypeSpec                [3]     IMPLICIT     FBTypeSpec}

SubrangeTypeSpec ::= SEQUENCE{  
           baseType     TypeID,     -- NOTE     minValue and maxValue  
           minValue     FixedLengthInteger,     -- shall be within the range  
           maxValue     FixedLengthInteger}     -- of baseType values

EnumeratedTypeSpec ::= SEQUENCE OF AnyString

```

BitNameDefintion ::= SEQUENCE {
    bitName  AnyString,
    bitNumber  USINT}

FBTypeSpec ::= SET{
    inputs          [0] IMPLICIT FbtElementSpec OPTIONAL,
    outputs         [1] IMPLICIT FbtElementTypeSpec OPTIONAL,
    controlInputs   [2] IMPLICIT FbtElementTypeSpec OPTIONAL,
    controlOutputs  [3] IMPLICIT FbtElementTypeSpec OPTIONAL}

FbtElementTypeSpec ::= SEQUENCE OF ElementSpec

ElementSpec ::= SEQUENCE {
    name      AnyString,
    typespec  ElementTypeSpec}

ElementTypeSpec ::= CHOICE {
    [0] IMPLICIT TypeID,
    [1] IMPLICIT SubrangeTypeSpec,
    [2] IMPLICIT EnumeratedTypeSpec,
    [3] IMPLICIT FormalArrayTypeSpec,
    [4] IMPLICIT ExpandedStringTypeSpec,
    [5] IMPLICIT ExpandedString2TypeSpec}

```

The following END statement shall terminate the ASN.1 module opened in 4.1.

END.

### 4.3 Encapsulation abstract syntax

#### 4.3.1 Encapsulation protocol

##### 4.3.1.1 General

In order to send Type 2 TPDU's over TCP/IP and UDP, an encapsulation protocol is required. Subclause 4.3 defines the encapsulation protocol requirements. The encapsulation messages are defined such that they are independent of the underlying transport (TCP or UDP), with a few exceptions as indicated.

Clause 11 specifies how the encapsulation protocol is used to transport Type 2 TPDU's over TCP/IP or UDP for explicit and implicit communications.

##### 4.3.1.2 TCP and UDP port numbers

The TCP port numbers defined in Table 207 are reserved with the Internet Assigned Numbers Authority (IANA) for use with the Type 2 Ethernet encapsulation protocol.

**Table 207 – TCP port numbers**

| Port Number                                                   | Description                                                                                         |
|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| 44818 (0xAF12)                                                | Used for non-secure encapsulation of TPDU's                                                         |
| 2221 (0x08AD)                                                 | Used in conjunction with TLS to provide a secure transport for encapsulation messages. <sup>a</sup> |
| <sup>a</sup> This usage is outside the scope of this document |                                                                                                     |

The UDP port numbers defined in Table 208 are reserved with the Internet Assigned Numbers Authority (IANA) for use with the Type 2 Ethernet encapsulation protocol.

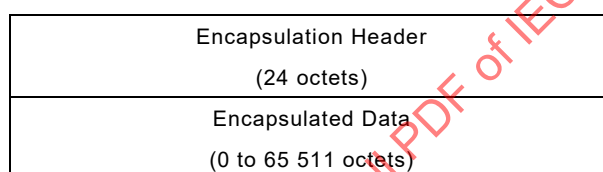
**Table 208 – UDP port numbers**

| Port Number                                                    | Description                                                                                                                                                                    |
|----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 44818 (0xAF12)                                                 | Used for non-secure encapsulation TPDUs and secure and non-secure UCMM and class 3 TPDUs                                                                                       |
| 2221 (0x08AD)                                                  | Used in conjunction with DTLS to provide a secure transport for Forward_Open / Forward_Close, transport class 0 and class 1 TPDUs using the Common Packet Format. <sup>a</sup> |
| 2222 (0x08AE)                                                  | Used Type 2 transport class 0 and class 1 TPDUs using the Common Packet Format.                                                                                                |
| <sup>a</sup> This usage is outside the scope of this document. |                                                                                                                                                                                |

## 4.3.2 Encapsulation messages

### 4.3.2.1 Encapsulation message structure

All encapsulation messages shall be composed of a fixed-length header of 24 octets followed by an optional data portion. The total encapsulation message length (including header) shall be limited to 65 535 octets. The encapsulation message length shall not override length restrictions imposed by the encapsulated protocol. Its structure shall be as shown in Figure 14, where the top of the diagram indicates the portion of the message sent first on the wire.

**Figure 14 – Encapsulation message**

The encapsulated data portion of the message is command specific and is required only for certain commands.

Multi-octet integer fields in encapsulation messages shall be transmitted as specified in 4.3.4.4 (i.e. using little endian octet ordering).

NOTE This is different from the octet ordering used in standard Internet network protocols, which is big endian.

### 4.3.2.2 Encapsulation header

The encapsulation header shall be as shown in Table 209.

**Table 209 – Encapsulation header**

| Field name     | Format            | Description                                                                                                             |
|----------------|-------------------|-------------------------------------------------------------------------------------------------------------------------|
| Command        | UINT              | Encapsulation command                                                                                                   |
| Length         | UINT              | Length, in octets, of the command specific data portion of the message, i.e., the number of octets following the header |
| Session handle | UDINT             | Session identification (application dependent)                                                                          |
| Status         | UDINT             | Status code                                                                                                             |
| Sender context | ARRAY of 8 USHORT | Information pertinent only to the sender of an encapsulation command                                                    |
| Options        | UDINT             | Options flags                                                                                                           |

Although the header contains no explicit information to distinguish between a request and a reply, this information shall be determined in either of two ways:

- implicitly, by the command and the context in which the message is generated. (For example, in the case of the RegisterSession command, the request is generated by an originator and the target generates the reply);
- explicitly, by the contents of an encapsulated protocol packet in the data part of the message.

#### 4.3.2.3 Command field

The allocation of command codes shall be as shown in Table 210.

**Table 210 – Encapsulation command codes**

| Command code                                                                                                                                                                                                                                                                                                                                                               | Name                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| 0x0000                                                                                                                                                                                                                                                                                                                                                                     | NOP                                         |
| 0x0001 to 0x0003                                                                                                                                                                                                                                                                                                                                                           | Reserved for legacy usage <sup>a</sup>      |
| 0x0004                                                                                                                                                                                                                                                                                                                                                                     | ListServices                                |
| 0x0005                                                                                                                                                                                                                                                                                                                                                                     | Reserved for legacy usage <sup>a</sup>      |
| 0x0006 to 0x0062                                                                                                                                                                                                                                                                                                                                                           | Reserved for future extensions <sup>b</sup> |
| 0x0063                                                                                                                                                                                                                                                                                                                                                                     | ListIdentity                                |
| 0x0064                                                                                                                                                                                                                                                                                                                                                                     | ListInterfaces                              |
| 0x0065                                                                                                                                                                                                                                                                                                                                                                     | RegisterSession                             |
| 0x0066                                                                                                                                                                                                                                                                                                                                                                     | UnRegisterSession                           |
| 0x0067 to 0x006E                                                                                                                                                                                                                                                                                                                                                           | Reserved for legacy usage <sup>a</sup>      |
| 0x006F                                                                                                                                                                                                                                                                                                                                                                     | SendRRData                                  |
| 0x0070                                                                                                                                                                                                                                                                                                                                                                     | SendUnitData                                |
| 0x0071 to 0x00C7                                                                                                                                                                                                                                                                                                                                                           | Reserved for legacy usage <sup>a</sup>      |
| 0x00C8                                                                                                                                                                                                                                                                                                                                                                     | StartDTLS <sup>c</sup>                      |
| 0x00C9 to 0xFFFF                                                                                                                                                                                                                                                                                                                                                           | Reserved for future extensions <sup>b</sup> |
| <sup>a</sup> Commands marked as "Reserved for legacy usage" indicate commands that were defined prior to the publication of this document. Their behavior is undefined in this document. Devices shall not implement these commands without prior knowledge of the legacy usage. Devices that do not support these commands shall return encapsulation status code 0x0001. |                                             |
| <sup>b</sup> Commands marked as "Reserved for future extensions" shall not be used.                                                                                                                                                                                                                                                                                        |                                             |
| <sup>c</sup> This command is outside the scope of this document.                                                                                                                                                                                                                                                                                                           |                                             |

If the encapsulation transport is TCP; a device shall accept commands that it does not support without breaking the session or underlying TCP connection. A status code indicating that an unsupported command was received shall be returned to the sender of the message (see 4.3.2.6).

The IP-layer transport (TCP or UDP) used for each of the encapsulation commands in Table 210 is dependent on the Type 2 Ethernet transport profile(s) the device implements.

NOTE Type 2 Ethernet transport profiles are defined in the Type 2 Ethernet Transports implementation profile, which is outside the scope of this document.

#### 4.3.2.4 Length field

The length field in the header shall specify the size in octets of the data portion of the message. The field shall contain zero for messages that contain no data. The total length of a message shall be the sum of the number contained in the length field plus the 24-octet size of the encapsulation header.

When using TCP, the entire encapsulation message shall be read from the TCP/IP connection even if the length is invalid for a particular command or exceeds the host's internal buffers. Data that would exceed internal buffers may be discarded, however the entire encapsulation messages shall be read.

NOTE Failure to read the entire message can result in losing track of the message boundaries in the TCP octet stream.

#### 4.3.2.5 Session handle field

The Session Handle shall be generated by the target and returned to the originator in response to a RegisterSession request. The originator shall insert it in all subsequent encapsulation command requests (sent using the commands listed in Table 210) which require sessions to that particular target. In the case where the target initiates and sends a command to the originator, the target shall include this field in the request that it sends to the originator.

NOTE Some commands (e.g. Nop) do not require a session handle, even if a session has been established. The specification of a particular command will indicate if it does not require a session.

#### 4.3.2.6 Status field

The value in the Status field shall indicate whether or not the receiver was able to execute the requested encapsulation command. A value of zero in a reply shall indicate successful execution of the command. In all requests issued by the sender, the Status field shall contain zero. If the receiver receives a request with a non-zero Status field, the request shall be ignored and no reply shall be generated.

NOTE This field does not reflect errors that are generated by an encapsulated protocol packet contained within the data portion of the message. For example, an error encountered during an end node's processing of a Set Attributes service would be returned via the Type 2 specified error mechanism.

The status codes shall be as shown in Table 211.

**Table 211 – Encapsulation status codes**

| Status code    | Description                                                                                                                                                                                                |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x00           | Success                                                                                                                                                                                                    |
| 0x01           | The sender issued an invalid or unsupported encapsulation command.                                                                                                                                         |
| 0x02           | Insufficient memory resources in the receiver to handle the command.<br>This is not an application error. Instead, it only occurs if the encapsulation layer cannot obtain memory resources that it needs. |
| 0x03           | Poorly formed or incorrect data in the data portion of the encapsulation message.                                                                                                                          |
| 0x04 to 0x63   | Reserved for legacy usage <sup>a</sup>                                                                                                                                                                     |
| 0x64           | An originator used an invalid session handle when sending an encapsulation message to the target.                                                                                                          |
| 0x65           | The target received a message of invalid length (see 4.3.2.4).                                                                                                                                             |
| 0x66 to 0x68   | Reserved for legacy usage <sup>a</sup>                                                                                                                                                                     |
| 0x69           | Unsupported protocol version.                                                                                                                                                                              |
| 0x6A           | Encapsulated Type 2 service not allowed on this port                                                                                                                                                       |
| 0x6B to 0xFFFF | Reserved for future extensions <sup>b</sup>                                                                                                                                                                |

|              |                                                                                                                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <sup>a</sup> | Status codes marked as "Reserved for legacy usage" indicate status codes that were defined prior to the publication of this document. Their usage is undefined in this document. Devices shall not use these status codes without prior knowledge of the legacy usage. |
| <sup>b</sup> | Status codes marked as "Reserved for future extensions" shall not be used.                                                                                                                                                                                             |

#### 4.3.2.7 Sender context field

The sender of the command shall assign the value in the Sender Context field of the header. The receiver shall return this value without modification in its reply. Commands with no expected reply may ignore this field.

The sender of a command request may place any value in this field.

NOTE This field can be used to match requests with their associated replies.

#### 4.3.2.8 Options field

The intent of this field is to provide the bits that modify the meaning of the various encapsulation commands. Options and option behavior are defined on a per-command basis.

#### 4.3.2.9 Command specific data field

The structure of the command specific data field depends on the command code. To organize their command specific data field, most commands use either or both of the following two methods:

- use a fixed structure;
- use the common packet format (specified in 4.3.4).

The common packet format allows commands to structure their command specific data field in an extensible way.

### 4.3.3 Command descriptions

#### 4.3.3.1 Nop

Either an originator or a target may send a Nop request. No reply shall be generated. The data portion of the request shall be from 0 to 65 511 octets long. The receiver shall ignore any data that is contained in the message. A Nop request does not require that a session be established.

NOTE A Nop provides a way for either an originator or target to determine if the TCP connection is still open.

The Nop request encapsulation header shall be as shown in Table 212.

**Table 212 – Nop request encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------|
| Command                                                                                                                                                  | UINT              | Nop (0x00)                          |
| Length                                                                                                                                                   | UINT              | Length of the command specific data |
| Session handle                                                                                                                                           | UDINT             | Any value (ignored by target)       |
| Status                                                                                                                                                   | UDINT             | 0 <sup>a</sup>                      |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Chosen by sender                    |
| Options                                                                                                                                                  | UDINT             | 0                                   |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                     |

### 4.3.3.2 RegisterSession

#### 4.3.3.2.1 General

An originator shall send a RegisterSession request to a target to initiate a session. The RegisterSession command does not require that a session be established.

NOTE See 11.7 for detailed information on establishing and maintaining a session.

#### 4.3.3.2.2 Request

The RegisterSession request encapsulation header shall be as shown in Table 213.

**Table 213 – RegisterSession request encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------|
| Command                                                                                                                                                  | UINT              | RegisterSession (0x65)        |
| Length                                                                                                                                                   | UINT              | 4                             |
| Session handle                                                                                                                                           | UDINT             | Any value (ignored by target) |
| Status                                                                                                                                                   | UDINT             | 0                             |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Any value                     |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                               |

The parameters in the data portion shall determine the version of the protocol and any session options as shown in Table 214. The protocol version shall be set to 1. No option flags are currently defined, so the session options flags shall be set to 0.

NOTE This options flags field is not the same as the options flags in the encapsulation header.

**Table 214 – RegisterSession request data portion**

| Field            | Data type | Description                    |
|------------------|-----------|--------------------------------|
| Protocol version | UINT      | Requested protocol version (1) |
| Options flags    | UINT      | Session options (0)            |

#### 4.3.3.2.3 Reply

The target shall send a RegisterSession reply to indicate that it has registered the originator. The reply shall have the same format as the request as shown in Table 215.

**Table 215 – RegisterSession reply encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|------------------------------------|
| Command                                                                                                                                                  | UINT              | RegisterSession (0x65)             |
| Length                                                                                                                                                   | UINT              | 4                                  |
| Session handle                                                                                                                                           | UDINT             | Session handle generated by target |
| Status                                                                                                                                                   | UDINT             | 0                                  |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Value from request                 |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                     |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                    |

The Session Handle field of the header shall contain a target-generated identifier that the originator shall save and insert in the Session Handle field of the header for all subsequent requests to that target. This field shall be valid only if the Status field is zero (0).

The Sender Context field of the header shall contain the same values present in the original sender request.

If the originator was successfully registered with the target, the Status field shall be zero (0). If the originator was not successfully registered, the Status field shall contain the appropriate status code, as follows:

- status code 0x0001 shall be returned if the originator attempts to register more than 1 active session on the same TCP connection;
- status code 0x0002 shall be returned if the target does not have sufficient resources to register the originator;
- other status codes from Table 211 may be used as appropriate for general encapsulation related errors (e.g., poorly formed encapsulation message, invalid length);
- status code 0x0069 shall be returned for Protocol Version or Options mismatches, as described below.

The data portion of the reply shall have the same format as the request as shown in Table 216.

**Table 216 – RegisterSession reply data portion (successful)**

| Field            | Data type | Description                                                                                                                                         |
|------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Protocol Version | UINT      | Version from RegisterSession request if supported.<br>If the requested version is not supported, contains the highest version supported.            |
| Options          | UINT      | Options flags from RegisterSession request if supported.<br>If any requested Options flags are not supported, contains the supported Options flags. |

The Protocol Version field shall equal the requested version if the originator was successfully registered. If the target does not support the requested version of the protocol:

- the session shall not be created;
- the Status field shall be set to "unsupported encapsulation protocol" (0x0069);
- the target shall return the highest supported version in the Protocol Version field.

At present, no Options flags are defined. In order to support their future definition, targets shall check the value of the Options flags in the RegisterSession request. If all requested options are supported, the Options field in the reply shall contain the originator's requested value. If the target does not support the requested options:

- the session shall not be created;
- the Status field shall be set to "unsupported encapsulation protocol" (0x0069);
- the target shall return the options that it supports in the RegisterSession reply.

#### 4.3.3.3 UnRegisterSession

Either an originator or a target may send this request to terminate the session. The receiver shall initiate a close of the underlying TCP/IP connection when it receives this request. The session shall also be terminated when the transport connection between the originator and target is terminated. The receiver shall perform any other associated cleanup required on its end. There shall be no reply to this command, except in the event that the command is received via UDP. If the command is received via UDP, the receiver shall reply with encapsulation status code 0x0001 (invalid or unsupported command).

The UnregisterSession request format shall be as shown in Table 217.

**Table 217 – UnRegisterSession request encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------|
| Command                                                                                                                                                  | UINT              | UnRegisterSession (0x65)            |
| Length                                                                                                                                                   | UINT              | 4                                   |
| Session Handle                                                                                                                                           | UDINT             | Session handle from RegisterSession |
| Status                                                                                                                                                   | UDINT             | 0                                   |
| Sender Context                                                                                                                                           | ARRAY of 8 USHORT | Any value (ignored by target)       |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                      |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                     |

The receiver shall not reject the UnRegisterSession due to unexpected values in the encapsulation header (invalid Session Handle, non-zero Status, non-zero Options, or additional command data). In all cases the TCP connection shall be closed.

NOTE See 11.7.2.3 for more details about terminating a session.

#### 4.3.3.4 ListServices

##### 4.3.3.4.1 General

The ListServices request shall be sent to determine which encapsulation service classes the target device supports. The ListServices command does not require that a session be established.

NOTE Each service class has a unique type code, and an optional ASCII name.

##### 4.3.3.4.2 Request

The ListServices request encapsulation header shall be as shown in Table 218.

**Table 218 – ListServices request encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------|
| Command                                                                                                                                                  | UINT              | ListServices (0x04)           |
| Length                                                                                                                                                   | UINT              | 0                             |
| Session handle                                                                                                                                           | UDINT             | Any value (ignored by target) |
| Status                                                                                                                                                   | UDINT             | 0                             |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Chosen by sender              |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                               |

#### 4.3.3.4.3 Reply

The receiver shall reply with an encapsulation message consisting of the header and data, as shown in Table 219 and Table 220. The data portion of the reply shall provide the information on the services supported.

**Table 219 – ListServices reply encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------|
| Command                                                                                                                                                  | UINT              | ListServices (0x04)                 |
| Length                                                                                                                                                   | UINT              | Length of the command specific data |
| Session handle                                                                                                                                           | UDINT             | Any value (ignored by receiver)     |
| Status                                                                                                                                                   | UDINT             | 0                                   |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Value from request                  |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                      |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                     |

The data portion of the reply shall contain a 2-octet item count followed by an array of items describing the service(s) provided, as shown in Table 220.

**Table 220 – ListServices reply data portion (successful)**

| Field name   | Data type         | Description                                          |
|--------------|-------------------|------------------------------------------------------|
| Item count   | UINT              | Number of items to follow                            |
| Target items | STRUCT of         | Interface information                                |
|              | UINT              | Item ID                                              |
|              | UINT              | Item length                                          |
|              | UINT              | Version of encapsulated protocol (shall be set to 1) |
|              | UINT              | Capability Flags                                     |
|              | ARRAY of 16 USINT | Name of service                                      |

The Item ID shall identify the service class. One service class is defined, with type code 0x100 and name "Communications". This service class shall indicate that the device supports encapsulation of Type 2 PDU's. All devices that support Type 2 PDU encapsulation shall support the ListServices command and Communications service class.

NOTE See 4.3.4 for a description of items and a list of all reserved item codes.

The Version field shall indicate the version of the service supported by the target to help maintain compatibility between applications.

Each service shall have a different set of capability flags. Reserved flags shall be set to zero.

The Capability Flags, defined for the Communications service, shall be as shown in Table 221.

**Table 221 – Communications capability flags**

| Flag value                                                                                                                                                                                                                                                                                                                                                                         | Description                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Bits 0 to 4                                                                                                                                                                                                                                                                                                                                                                        | Reserved for legacy usage <sup>a</sup>                                                                          |
| Bit 5                                                                                                                                                                                                                                                                                                                                                                              | If the device supports Type 2 PDU encapsulation this bit shall be set (= 1); otherwise, it shall be clear (= 0) |
| Bits 6 to 7                                                                                                                                                                                                                                                                                                                                                                        | Reserved for legacy usage <sup>a</sup>                                                                          |
| Bit 8                                                                                                                                                                                                                                                                                                                                                                              | Supports transport class 0 or 1 UDP-based connections                                                           |
| Bits 9 to 15                                                                                                                                                                                                                                                                                                                                                                       | Reserved for future extensions                                                                                  |
| <sup>a</sup> Flag marked as "Reserved for legacy usage" indicate flags that were defined prior to the publication of this document. Their usage is undefined in this document. Devices shall not use these flags without prior knowledge of the legacy usage. If a device receives a reserved flag that it does not understand, the reply shall be processed and the flag ignored. |                                                                                                                 |

The Name field shall allow up to a 16-octet, NULL-terminated ASCII string for descriptive purposes only. The 16-octet limit shall include the NULL character.

#### **4.3.3.5 ListIdentity**

##### **4.3.3.5.1 General**

A connection originator may use the ListIdentity request to locate and identify potential targets. This request shall be sent as a unicast message using TCP or UDP, or as a broadcast message using UDP and does not require that a session be established. The reply shall always be sent as a unicast message.

When received as a broadcast message, the receiving device shall delay for a pseudo-random period of time prior to sending the reply as specified below. Delaying before sending the reply helps to spread out any resulting ARP requests and ListIdentity replies from target devices on the network.

##### **4.3.3.5.2 Request**

The ListIdentity request encapsulation header shall be as shown in Table 222.

**Table 222 – ListIdentity request encapsulation header**

| Field name     | Data type         | Field value                                                   |
|----------------|-------------------|---------------------------------------------------------------|
| Command        | UINT              | ListIdentity (0x63)                                           |
| Length         | UINT              | 0                                                             |
| Session handle | UDINT             | Any value (ignored by target)                                 |
| Status         | UDINT             | 0                                                             |
| Sender context | UINT              | MaxResponseDelay in ms, see below                             |
|                | ARRAY of 6 USHORT | Reserved, shall be ignored by the receiver, values shall be 0 |
| Options        | UDINT             | 0 <sup>a</sup>                                                |

<sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field.

#### 4.3.3.5.3 Reply

Several reply items are defined for this command. The CFP 2 Identity item shall be supported (returned) by all Type 2 devices.

A receiver of the ListIdentity command shall reply with an encapsulation message consisting of the header and applicable data items, as shown in Table 223 and Table 224. The reply shall be sent to the IP address from which the request was received.

When the ListIdentity request has been received as a UDP broadcast message, the receiver shall delay before sending the reply. When received as a unicast message (either via UDP or TCP), the receiver shall not delay.

The receiver's delay shall be a random value, in milliseconds, between 0 and the MaxResponseDelay specified in the ListIdentity request. If the sender specifies a MaxResponseDelay value of 0 ms, a default value of 2 000 ms shall be used by the receiver. If the sender specifies a MaxResponseDelay value of 1 ms to 500 ms, a value of 500 ms shall be used by the receiver. A new random value shall be chosen for each request.

The purpose of the delay is to spread the ListIdentity responses (and the ARP messages that can result) over the MaxResponseDelay interval. It is therefore important that each device generate a unique random delay value. Devices shall ensure they each generate a unique value, for example by seeding their random number generators with a unique value such as their IP address or Ethernet MAC address.

It is possible that devices receive additional broadcast ListIdentity requests while delaying for the response, for example, if multiple clients issue the broadcast ListIdentity request. Devices should be able to accept and process at least 2 outstanding broadcast ListIdentity requests concurrently.

After issuing a broadcast ListIdentity request, a client should not issue further requests until the MaxResponseDelay interval for its current request has expired. Client behavior for handling ListIdentity responses received beyond MaxResponseDelay is vendor specific.

**Table 223 – ListIdentity reply encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------|
| Command                                                                                                                                                  | UINT              | List Identity (0x63)                |
| Length                                                                                                                                                   | UINT              | Length of the command specific data |
| Session handle                                                                                                                                           | UDINT             | Any value (ignored by receiver)     |
| Status                                                                                                                                                   | UDINT             | 0                                   |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Value from request                  |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                      |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                     |

The data portion of the reply is structured as a Common Packet Format that contains a 2-octet item count followed by an array of items (see Table 224), providing the target identity as a minimum.

**Table 224 – ListIdentity reply data portion (successful)**

| Field name   | Data type       | Field value                      |
|--------------|-----------------|----------------------------------|
| Item count   | UINT            | Number of target items to follow |
| Target items | STRUCT of       |                                  |
|              | UINT            | Item ID                          |
|              | UINT            | Item length                      |
|              | ARRAY of octets | Item data                        |

The CFP 2 Identity item shall be the first item returned and has the format as defined in Table 225. Part of this item definition follows the Get Attribute All service response definition of the Identity object (data returned based on instance one of this object). Unlike most fields in the Common Packet Format, the Socket Address field shall be sent in big endian order.

Clients should be aware that, while the Socket Address information supplied in the Type 2 Identity item is accurate from the perspective of the device sending the response, it could be not accurate from the perspective of the client if address translation such as NAT is used between the client and the device.

Two additional items are defined, the Type 2 Security Information item and the Type 2 Ethernet Capability item.

NOTE 1 The Type 2 Security Information item is outside the scope of this document.

Receivers of the ListIdentity reply shall ignore unexpected items.

**Table 225 – Type 2 identity item**

| Field name                                                                                                                     | Data type      | Description                                                                                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Item ID                                                                                                                        | UINT           | Item ID of Type 2 Identity (0x0C)                                                                                                                                                                                              |
| Item length                                                                                                                    | UINT           | Number of octets in item which follows (length varies depending on Product Name string)                                                                                                                                        |
| Version                                                                                                                        | UINT           | Encapsulation protocol version supported (also returned with Register Session reply)                                                                                                                                           |
| Socket Address                                                                                                                 | STRUCT of:     | Socket Address describing the device's Encapsulation interface                                                                                                                                                                 |
| sin_family                                                                                                                     | INT            | Shall be AF_INET = 2.<br>This field shall be sent in big endian order.                                                                                                                                                         |
| sin_port                                                                                                                       | UINT           | Shall be 0xAF12 (TCP port number for the unsecured Encapsulation protocol).<br>This field shall be sent in big endian order. Notice that this port may be closed on secure devices, but shall still be returned in this field. |
| sin_addr                                                                                                                       | UDINT          | Shall be the IPv4 address of the IP interface in this device that is responding to the request.<br>This field shall be sent in big endian order.                                                                               |
| sin_zero                                                                                                                       | ARRAY of USINT | Length of 8. Should be set to zero by sender. Ignored by receiver.                                                                                                                                                             |
| Vendor ID <sup>a</sup>                                                                                                         | UINT           | Device manufacturers Vendor ID                                                                                                                                                                                                 |
| Device Type <sup>a</sup>                                                                                                       | UINT           | Device Type of product                                                                                                                                                                                                         |
| Product Code <sup>a</sup>                                                                                                      | UINT           | Product Code assigned with respect to device type                                                                                                                                                                              |
| Revision <sup>a</sup>                                                                                                          | USINT[2]       | Device revision                                                                                                                                                                                                                |
| Status <sup>a</sup>                                                                                                            | WORD           | Current status of device                                                                                                                                                                                                       |
| Serial Number <sup>a</sup>                                                                                                     | UDINT          | Serial number of device                                                                                                                                                                                                        |
| Product Name <sup>a</sup>                                                                                                      | SHORT_STRING   | Human readable description of device                                                                                                                                                                                           |
| State <sup>b</sup>                                                                                                             | USINT          | Current state of device                                                                                                                                                                                                        |
| <sup>a</sup> These parameters are further defined by the corresponding instance attribute of the Identity object.              |                |                                                                                                                                                                                                                                |
| <sup>b</sup> The State attribute is an optional attribute of the Identity Object. If not implemented, the value shall be 0xFF. |                |                                                                                                                                                                                                                                |

The Type 2 Ethernet Capability item has the format as defined in Table 226. It may be used to define Type 2 Ethernet transport capabilities for the various message types for a device.

**Table 226 – Type 2 Ethernet Capability item**

| Field name                        | Data type | Description                                                                                                                                                                                                                                        |
|-----------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Item ID                           | UINT      | Item ID of Type 2 Ethernet Capability (0x87)                                                                                                                                                                                                       |
| Item length                       | UINT      | Number of octets = 4                                                                                                                                                                                                                               |
| Type 2 Ethernet transport profile | DWORD     | This field provides a list of transport options available on TCP and UDP (UCMM, Type 2 class 0, 1, 2 and 3 connections).<br>It is equivalent to the Features Supported member definition of the Type 2 Ethernet Transports implementation profile. |

This item is required for devices that implement some Type 2 Ethernet transport profiles.

NOTE 2 Type 2 Ethernet transport profiles are defined in the Type 2 Ethernet Transports implementation profile, which is outside the scope of this document.

### 4.3.3.6 ListInterfaces

#### 4.3.3.6.1 General

The optional ListInterfaces request shall be used by a connection originator to identify non-Type 2 communication interfaces associated with the target. A session need not be established to send this command.

#### 4.3.3.6.2 Request

The ListInterfaces request encapsulation header shall be as shown in Table 227.

**Table 227 – ListInterfaces request encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------|
| Command                                                                                                                                                  | UINT              | List Interfaces (0x64)        |
| Length                                                                                                                                                   | UINT              | 0                             |
| Session handle                                                                                                                                           | UDINT             | Any value (ignored by target) |
| Status                                                                                                                                                   | UDINT             | 0                             |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Chosen by sender              |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                               |

#### 4.3.3.6.3 Reply

If supported, the receiver of a ListInterfaces request command shall reply with an encapsulation message consisting of the header and data, as shown below in Table 228.

**Table 228 – ListInterfaces reply encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------|
| Command                                                                                                                                                  | UINT              | List Interfaces (0x64)              |
| Length                                                                                                                                                   | UINT              | Length of the command specific data |
| Session handle                                                                                                                                           | UDINT             | Any value (ignored by receiver)     |
| Status                                                                                                                                                   | UDINT             | 0                                   |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Value from request                  |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                      |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                     |

The data portion of the reply contains an array of items providing interface information. It is structured as a Common Packet Format, which contains a 2-octet item count followed by an array of items.

At present no public items are defined for the ListInterfaces reply. If no items are included, the Item Count shall be set to 0.

Some legacy devices may return "Reserved for legacy usage items" (see 4.3.4). Such items shall be ignored unless the receiving device has explicit knowledge of the legacy format and usage.

### 4.3.3.7 SendRRData

#### 4.3.3.7.1 General

A SendRRData request shall transfer an encapsulated request/reply packet between the originator and target, where the originator initiates the command. The actual request/reply packets shall be encapsulated in the data portion of the message and shall be the responsibility of the target and originator.

NOTE When used to encapsulate Type 2 PDU's, the SendRRData request and reply are used to send encapsulated UCMM messages (see Clause 11).

#### 4.3.3.7.2 Request

The SendRRData request encapsulation header shall be as shown in Table 229.

**Table 229 – SendRRData request encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------|
| Command                                                                                                                                                  | UINT              | SendRRData (0x6F)                   |
| Length                                                                                                                                                   | UINT              | Length of the command specific data |
| Session handle                                                                                                                                           | UDINT             | TCP: Session handle<br>UDP: 0       |
| Status                                                                                                                                                   | UDINT             | 0                                   |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Chosen by sender                    |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                      |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                     |

The data portion of the message shall contain request parameters and the encapsulated protocol packet as shown in Table 230.

**Table 230 – SendRRData request data portion**

| Field name                   | Data type       | Description                                     |
|------------------------------|-----------------|-------------------------------------------------|
| Interface handle             | UDINT           | Shall be 0                                      |
| Timeout                      | UINT            | Operation Timeout (0 to 65 535)                 |
| Encapsulated protocol packet | ARRAY of octets | See Common Packet Format specification in 4.3.4 |

The Interface handle shall identify the Communications Interface to which the request is directed. This handle shall be 0 for encapsulating Type 2 PDUs.

The target shall abort the requested operation after the timeout expires. When the "Timeout" field is in the range 1 to 65 535, the timeout shall be set to this number of seconds. When the "Timeout" field is set to 0, the encapsulation protocol shall not have its own timeout. Instead, it shall rely on the timeout mechanism of the encapsulated protocol.

When the SendRRData command is used to encapsulate Type 2 PDU's, the Timeout field shall be set to 0, and shall be ignored by the target.

The encapsulated protocol packet shall be encoded in a Common Packet Format as shown in 4.3.4.

#### 4.3.3.7.3 Reply

The SendRRData reply, as shown in Table 231, shall contain data in response to the SendRRData request. The reply to the original encapsulated protocol request shall be contained in the data portion of the SendRRData reply.

**Table 231 – SendRRData reply encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------|
| Command                                                                                                                                                  | UINT              | SendRRData (0x6F)                   |
| Length                                                                                                                                                   | UINT              | Length of the command specific data |
| Session handle                                                                                                                                           | UDINT             | Value from request                  |
| Status                                                                                                                                                   | UDINT             | 0                                   |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Value from request                  |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                      |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                                     |

The format of the data portion of the reply message shall be the same as that of the SendRRData request message. Since the request and reply share a common format, the reply message contains a Timeout field; however, it is not used.

#### 4.3.3.8 SendUnitData

The SendUnitData request shall send encapsulated connected messages. A reply need not be returned. This command may be used when the encapsulated protocol has its own underlying end-to-end transport mechanism. The SendUnitData request may be sent by either end of the TCP connection.

NOTE When used to encapsulate Type 2 PDU's, the SendUnitData command is used to send Type 2 connected data in both the O⇒T and T⇒O directions.

The format of the SendUnitData request shall be as shown in Table 232.

**Table 232 – SendUnitData request encapsulation header**

| Field name                                                                                                                                               | Data type         | Field value                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------|
| Command                                                                                                                                                  | UINT              | SendUnitData (0x70)           |
| Length                                                                                                                                                   | UINT              | Length of data portion        |
| Session handle                                                                                                                                           | UDINT             | TCP: Session handle<br>UDP: 0 |
| Status                                                                                                                                                   | UDINT             | 0                             |
| Sender context                                                                                                                                           | ARRAY of 8 USHORT | Any value (ignored by target) |
| Options                                                                                                                                                  | UDINT             | 0 <sup>a</sup>                |
| <sup>a</sup> For backwards compatibility, no options shall be defined for this command. The receiver shall discard packets with a non-zero option field. |                   |                               |

The data portion of the message shall contain request parameters as well as the encapsulated protocol packet as shown in Table 233.

**Table 233 – SendUnitData request data portion**

| Field name                   | Data type       | Description                                     |
|------------------------------|-----------------|-------------------------------------------------|
| Interface handle             | UDINT           | Shall be 0                                      |
| Timeout                      | UINT            | Operation timeout (shall be 0)                  |
| Encapsulated protocol packet | ARRAY of octets | See Common Packet Format specification in 4.3.4 |

Interface handle and Timeout shall be set to zero. The timeout field is not used since no reply is generated upon receipt of a SendUnitData command.

#### 4.3.4 Common packet format

##### 4.3.4.1 General

The common packet format (CPF) defines a standard format for protocol packets that are transported with the encapsulation protocol. The common packet format is a general-purpose mechanism designed to accommodate future packet or address types.

The common packet format shall consist of an item count, followed by a number of items (see Table 234). Some items are classified as "address items" (carries addressing information) or "data items" (carries encapsulated data). The number of items to be included depends on the encapsulation command and usage of the command. Subclause 4.3.4.4 specifies the valid Common Packet Format items for the various commands and usages.

**Table 234 – Common packet format**

| Field name | Data type               | Description               |
|------------|-------------------------|---------------------------|
| Item count | UINT                    | Number of items to follow |
| Item #1    | Item Struct (see below) | First CPF item            |
| Item #2    | Item Struct (see below) | Second CPF item           |
| ...        | ...                     | ...                       |
| Item #n    | Item Struct (see below) | n <sup>th</sup> CPF item  |

The address and data item structure shall be as shown in Table 235.

**Table 235 – CPF item format**

| Field name | Data type | Description                        |
|------------|-----------|------------------------------------|
| Type ID    | UINT      | Type of item encapsulated          |
| Length     | UINT      | Length in octets of data to follow |
| Data       | Variable  | The data (if length >0)            |

The item type ID numbers shall be as shown in Table 236.

**Table 236 – Item Type ID numbers**

| Item ID number                                                                                                                                                                                                                                                                                                                                                                                                                           | Item type | Description                                                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x0000                                                                                                                                                                                                                                                                                                                                                                                                                                   | address   | Null (used for UCMM messages). Indicates that encapsulation routing is NOT needed. Target is either local (Ethernet) or routing info is in a data item. |
| 0x0001 to 0x000B                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for legacy usage <sup>a</sup>                                                                                                                  |
| 0x000C                                                                                                                                                                                                                                                                                                                                                                                                                                   |           | Type 2 Identity (see 4.3.3.5.3 for item definition)                                                                                                     |
| 0x000D to 0x0085                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for legacy usage <sup>a</sup>                                                                                                                  |
| 0x0086                                                                                                                                                                                                                                                                                                                                                                                                                                   |           | Type 2 Security Information <sup>c</sup>                                                                                                                |
| 0x0087                                                                                                                                                                                                                                                                                                                                                                                                                                   |           | Type 2 Ethernet Capability (see 4.3.3.5.3 for item definition)                                                                                          |
| 0x0088 to 0x0090                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for future extensions <sup>b</sup>                                                                                                             |
| 0x0091                                                                                                                                                                                                                                                                                                                                                                                                                                   |           | Reserved for legacy usage <sup>a</sup>                                                                                                                  |
| 0x0092 to 0x00A0                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for future extensions <sup>b</sup>                                                                                                             |
| 0x00A1                                                                                                                                                                                                                                                                                                                                                                                                                                   | address   | Connection-based (used for connected messages)                                                                                                          |
| 0x00A2 to 0x00A4                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for legacy usage <sup>a</sup>                                                                                                                  |
| 0x00A5 to 0x00B0                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for future extensions <sup>b</sup>                                                                                                             |
| 0x00B1                                                                                                                                                                                                                                                                                                                                                                                                                                   | data      | Connected Transport packet                                                                                                                              |
| 0x00B2                                                                                                                                                                                                                                                                                                                                                                                                                                   | data      | Unconnected message                                                                                                                                     |
| 0x00B3 to 0x00FF                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for future extensions <sup>b</sup>                                                                                                             |
| 0x0100                                                                                                                                                                                                                                                                                                                                                                                                                                   |           | ListServices response (see 4.3.3.4.3 for item definition)                                                                                               |
| 0x0101 to 0x010F                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for legacy usage <sup>a</sup>                                                                                                                  |
| 0x0110 to 0x7FFF                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for future extensions <sup>b</sup>                                                                                                             |
| 0x8000                                                                                                                                                                                                                                                                                                                                                                                                                                   | data      | Sockaddr Info, originator-to-target                                                                                                                     |
| 0x8001                                                                                                                                                                                                                                                                                                                                                                                                                                   | data      | Sockaddr Info, target-to-originator                                                                                                                     |
| 0x8002                                                                                                                                                                                                                                                                                                                                                                                                                                   |           | Sequenced Address item                                                                                                                                  |
| 0x8003 to 0xFFFF                                                                                                                                                                                                                                                                                                                                                                                                                         |           | Reserved for future extensions <sup>b</sup>                                                                                                             |
| <sup>a</sup> Items marked as "Reserved for legacy usage" indicate Item IDs that were defined prior to the publication of this document. Their behavior is undefined in this document. Devices shall not use these Item IDs without prior knowledge of the legacy usage.<br><sup>b</sup> Products compliant with this specification shall not use command codes in this range<br><sup>c</sup> This is outside the scope of this document. |           |                                                                                                                                                         |

#### 4.3.4.2 Address items

##### 4.3.4.2.1 Null

The null address item shall contain only the type id and the length as shown in Table 237. The length shall be zero. No data shall follow the length. Since the null address item contains no routing information, it shall be used when the protocol packet itself contains any necessary routing information. The null address item shall be used for Unconnected Messages.

**Table 237 – Null address item**

| Field name | Data type | Field value |
|------------|-----------|-------------|
| Type ID    | UINT      | 0           |
| Length     | UINT      | 0           |

#### 4.3.4.2.2 Connected

This address item shall be used when the encapsulated protocol is connection-oriented. The data shall contain a connection identifier, as shown in Table 238.

NOTE Connection identifiers are exchanged in the Forward\_Open service of the Connection Manager.

**Table 238 – Connected address item**

| Field name | Data type | Field value           |
|------------|-----------|-----------------------|
| Type ID    | UINT      | 0xA1                  |
| Length     | UINT      | 4                     |
| Data       | UDINT     | Connection Identifier |

#### 4.3.4.2.3 Sequenced address item

This address item shall be used for class 0 and class 1 connected data. The data shall contain a connection identifier and an Encapsulation Sequence Number, as shown in Table 239. Usage is further described in 11.3.3.

**Table 239 – Sequenced address item**

| Field name | Data type | Field value                   |
|------------|-----------|-------------------------------|
| Type ID    | UINT      | 0x8002                        |
| Length     | UINT      | 8                             |
| Data       | UDINT     | Connection Identifier         |
|            | UDINT     | Encapsulation Sequence Number |

#### 4.3.4.3 Data items

##### 4.3.4.3.1 Unconnected

The data item that encapsulates an unconnected message shall be as shown in Table 240.

**Table 240 – Unconnected data item**

| Field name | Data type | Field value                                   |
|------------|-----------|-----------------------------------------------|
| Type ID    | UINT      | 0xB2                                          |
| Length     | UINT      | Length, in octets, of the unconnected message |
| Data       | Variable  | The unconnected message                       |

The format of the "data" field is dependent on the encapsulated protocol. When used to encapsulate Type 2 PDU's, the format of the "data" field shall be the same as PDU destined for the Message Router. The UCMM header, defined in 4.1.3, shall not be present. The context field in the encapsulation header shall be used for unconnected request/reply matching.

##### 4.3.4.3.2 Connected

The data item that encapsulates a connected transport PDU shall be as shown in Table 241.

**Table 241 – Connected data item**

| Field name | Data type | Field value                             |
|------------|-----------|-----------------------------------------|
| Type ID    | UINT      | 0xB1                                    |
| Length     | UINT      | Length, in octets, of the transport PDU |
| Data       | Variable  | The transport PDU                       |

The format of the "data" field is dependent on the encapsulated protocol. When used to encapsulate Type 2 PDU's, the format of the "data" field shall be the same as the connected transport PDU.

#### 4.3.4.3.3 Sockaddr info

The Sockaddr Info items shall be used to communicate IP address or port information necessary to create Class 0 or Class 1 connections. There are separate items for originator-to-target and target-to-originator socket information. The items are present as additional data in Forward\_Open / Large\_Forward\_Open request and reply services encapsulated in a SendRRData message. Subclause 11.2.4 specifies the usage of these items in the context of creating Type 2 connections.

The Sockaddr Info items shall have the structure shown in Table 242.

**Table 242 – Sockaddr info items**

| Field name | Data type        | Field value                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type ID    | UINT             | 0x8000 for O⇒T, 0x8001 for T⇒O                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Length     | UINT             | 16 (octets)                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| sin_family | INT              | Shall be AF_INET = 2.<br>This field shall be sent in big endian order                                                                                                                                                                                                                                                                                                                                                                               |
| sin_port   | UINT             | For point-point connections, sin_port shall be set to the UDP port to which packets for this Type 2 connection will be sent. For point-point connections, it is recommended that the registered UDP port (0x8AE) be used.<br><br>When used with a multicast connection, the sin_port field shall be set to the registered UDP port number (0x08AE) and treated by the receiver as "don't care".<br><br>This field shall be sent in big endian order |
| sin_addr   | UDINT            | For multicast connections, sin_addr shall be set to the IP multicast address to which packets for this Type 2 connection will be sent. When used with a point-point connection, the sin_addr field shall be treated by the receiver as "don't care". It is recommended that the sender set sin_addr to 0 for point-point connections.<br><br>This field shall be sent in big endian order.                                                          |
| sin_zero   | ARRAY of 8 USINT | Recommended value of zero; not enforced                                                                                                                                                                                                                                                                                                                                                                                                             |

The format of the Sockaddr Info item has been patterned after the sockaddr\_in structure as defined in the Winsock specification, version 1.1.

#### 4.3.4.4 Valid Common Packet Format item usage summary

The Common Packet Format is used with the following encapsulation commands:

- ListIdentity reply
- ListInterfaces reply
- ListServices reply
- SendRRData request and reply
- SendUnitData
- Transport class 0 and class 1 packets (no encapsulation header)

Table 243 shows the valid usage of CPF items for the various encapsulation commands.

NOTE Clause 11 contains the detailed formats and usage for Type 2 connected and unconnected messages.

**Table 243 – Usage of CPF items**

| Command              | Required CPF items                                 | Optional CPF items                                                                                                           | Action if unexpected or undefined CPF items are present                                                                   |
|----------------------|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| ListIdentity reply   | Type 2 Identity item                               | Type 2 Security Information<br>Type 2 Ethernet Capability                                                                    | Ignore items                                                                                                              |
| ListInterfaces reply | None                                               | Legacy devices may return "Reserved for legacy usage" items                                                                  | Ignore items                                                                                                              |
| ListServices reply   | ListServices item for the "Communications" service | Legacy devices may return "Reserved for legacy usage" items                                                                  | Ignore items                                                                                                              |
| SendRRData request   | Address item followed by Data item                 | When used to encapsulate the Forward_Open service, additional Sockaddr_info items can be present, as specified in Clause 11  | Return error (0x0003)                                                                                                     |
| SendRRData reply     | Address item followed by Data item                 | When used to encapsulate the Forward_Open response, additional Sockaddr_info items can be present, as specified in Clause 11 | Signal error to the calling application. For Forward_Open response, connection shall not be established at the originator |
| SendUnitData         | Address item followed by Data item                 | None                                                                                                                         | Discard message                                                                                                           |
| Class 0/1 packet     | Address item followed by Data item                 | None                                                                                                                         | Discard message                                                                                                           |

## 5 Transfer syntax

### 5.1 Compact encoding

#### 5.1.1 Encoding rules

Subclause 5.1 describes the means by which the data types defined in this document shall be encoded/transferred across the control network. The abstract syntax definition along with a particular set of encoding rules shall result in the transfer syntax. For application user data, a single set of encoding rules shall be defined (Compact Encoding), resulting in the Compact transfer syntax.

Compact Encoding rules shall start with the encoding rules defined in ISO/IEC 8825-1. Compact Encoding then applies optimization rules, starting with the outer most Service Data Unit (SDU) and progressing to each successive encapsulated SDU. Compact Encoding shall define a more efficient encoding mechanism by reducing the amount of information (overhead) transferred between devices.

The difference between a Compact encoded value and an ASN.1 encoded value shall be the removal of the fields describing the type and length of the information. The TAG and LENGTH components of an ASN.1-encoded value shall not be transmitted on the control network. In addition, the Compact Encoding rules shall indicate that octet ordering rules are the reverse of those seen in ASN.1.

Given the conditions listed in 5.1.2, general rules shall be applied to an ASN.1 encoded value to generate a Compact encoded value. The general rules shall be as follows:

- remove the Identifier Octets;
- remove the "TAG" octets specified by ASN.1;
- remove the Length Octets;
- remove the "LENGTH" octets specified by ASN.1;
- reverse the octet ordering for multiple content octets.

### 5.1.2 Encoding constraints

The representation of a variable value using Compact Encoding shall be possible with the following restrictions:

- the variable type shall be fixed length and shall have no conditional or optional fields;
- the encoding of a given variable shall be represented with a constant number of octets derived from the type specification of this variable.

### 5.1.3 Examples

#### 5.1.3.1 BOOL encoding

The BOOLEAN encoding shall be performed on a single OCTET as described in Table 244.

**Table 244 – BOOLEAN encoding**

| If the value is: | Then:                   |
|------------------|-------------------------|
| FALSE            | The octet shall be 0x00 |
| TRUE             | The octet shall be 0x01 |

A FALSE BOOL shall be represented as shown in Table 245.

**Table 245 – Example compact encoding of a BOOL value**

| Octet number | 1 <sup>st</sup> |
|--------------|-----------------|
| BOOL         | 00              |

#### 5.1.3.2 SignedInteger encoding

The SignedInteger encoding shall be performed as described in Table 246.

**Table 246 – Encoding of SignedInteger values**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> | 5 <sup>th</sup> | 6 <sup>th</sup> | 7 <sup>th</sup> | 8 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| SINT         | 0LSB            |                 |                 |                 |                 |                 |                 |                 |
| INT          | 0LSB            | 1LSB            |                 |                 |                 |                 |                 |                 |
| DINT         | 0LSB            | 1LSB            | 2LSB            | 3LSB            |                 |                 |                 |                 |
| LINT         | 0LSB            | 1LSB            | 2LSB            | 3LSB            | 4LSB            | 5LSB            | 6LSB            | 7LSB            |

NOTE The example in Table 247 illustrates the encoding of a variable of type DINT whose value is 0x12345678.

**Table 247 – Example compact encoding of a SignedInteger value**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|
| DINT         | 78              | 56              | 34              | 12              |

### 5.1.3.3 UnsignedInteger encoding

The UnsignedInteger encoding shall be performed as described in Table 248.

**Table 248 – UnsignedInteger values**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> | 5 <sup>th</sup> | 6 <sup>th</sup> | 7 <sup>th</sup> | 8 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| USINT        | 0LSB            |                 |                 |                 |                 |                 |                 |                 |
| UINT         | 0LSB            | 1LSB            |                 |                 |                 |                 |                 |                 |
| UDINT        | 0LSB            | 1LSB            | 2LSB            | 3LSB            |                 |                 |                 |                 |
| ULINT        | 0LSB            | 1LSB            | 2LSB            | 3LSB            | 4LSB            | 5LSB            | 6LSB            | 7LSB            |

NOTE Table 249 illustrates the encoding of a variable of type UDINT whose value is 0xAABBCCDD.

**Table 249 – Example compact encoding of an UnsignedInteger**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|
| UDINT        | DD              | CC              | BB              | AA              |

### 5.1.3.4 FixedLengthReal encoding

The FixedLengthReal encoding shall be performed as described in Table 250.

**Table 250 – FixedLengthReal values**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> | 5 <sup>th</sup> | 6 <sup>th</sup> | 7 <sup>th</sup> | 8 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| REAL         | 0LSB            | 1LSB            | 2LSB            | 3LSB            |                 |                 |                 |                 |
| LREAL        | 0LSB            | 1LSB            | 2LSB            | 3LSB            | 4LSB            | 5LSB            | 6LSB            | 7LSB            |

Table 251 illustrates the encoding of a variable (Float1) whose type is REAL and whose value is Float1: = 10,0.

NOTE 1 The assignment of the value is using the IEC 61131-3 notation. The value is 0x41200000 in IEEE format:  $+1 \times 1,25 \times 2^3$ , sign bit is 0 (+1), exponent is 130 (bias 127), fraction is 0,25.

**Table 251 – Example compact encoding of a REAL value**

| Octet contents | 0LSB | 1LSB | 2LSB | 3LSB |
|----------------|------|------|------|------|
| REAL           | 00   | 00   | 20   | 41   |

Table 252 illustrates the encoding of a variable (Float2) whose type is LREAL and whose value is Float2: = –100,0.

NOTE 2 The value is 0xC059000000000000 in IEEE format:  $-1 \times 1,5625 \times 2^6$ , sign bit is 1 (-1), exponent is 1 029 (bias 1 023), fraction is 0,5625.

**Table 252 – Example compact encoding of a LREAL value**

| Octet contents | 0LSB | 1LSB | 2LSB | 3LSB | 4LSB | 5LSB | 6LSB | 7LSB |
|----------------|------|------|------|------|------|------|------|------|
| LREAL          | 00   | 00   | 00   | 00   | 00   | 00   | 59   | C0   |

### 5.1.3.5 Time encodings

Subclause 5.1.3.5 gives examples of the Compact Encoding of TIME, DATE, TIME\_OF\_DAY, DATE\_AND\_TIME, FTIME, LTIME, ITIME, NTIME, STIME, UTIME data values. Table 253 provides a generic illustration of the encoding of Time values.

**Table 253 – FixedLengthReal values**

| Octet number  | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> | 5 <sup>th</sup> | 6 <sup>th</sup> | 7 <sup>th</sup> | 8 <sup>th</sup> |
|---------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| TIME          | 0LSB            | 1LSB            | 2LSB            | 3LSB            |                 |                 |                 |                 |
| DATE          | 0LSB            | 1LSB            |                 |                 |                 |                 |                 |                 |
| TIME_OF_DAY   | 0LSB            | 1LSB            | 2LSB            | 3LSB            |                 |                 |                 |                 |
| DATE_AND_TIME | 0LSB-<br>Time   | 1LSB-<br>Time   | 2LSB-<br>Time   | 3LSB-<br>Time   | 0LSB-<br>Date   | 1LSB-<br>Date   |                 |                 |
| FTIME         | 0LSB            | 1LSB            | 2LSB            | 3LSB            |                 |                 |                 |                 |
| LTIME         | 0LSB            | 1LSB            | 2LSB            | 3LSB            | 4LSB            | 5LSB            | 6LSB            | 7LSB            |
| ITIME         | 0LSB            | 1LSB            |                 |                 |                 |                 |                 |                 |
| NTIME         | 0LSB            | 1LSB            | 2LSB            | 3LSB            | 4LSB            | 5LSB            | 6LSB            | 7LSB            |
| STIME         | 0LSB            | 1LSB            | 2LSB            | 3LSB            | 4LSB            | 5LSB            | 6LSB            | 7LSB            |
| UTIME         | 0LSB            | 1LSB            | 2LSB            | 3LSB            | 4LSB            | 5LSB            | 6LSB            | 7LSB            |

### 5.1.3.6 String encodings

Subclause 5.1.3.6 gives examples of the Compact Encoding of STRING, STRING2, STRINGN, and SHORT\_STRING data values.

NOTE The preferred string type for user supplied string data is STRING2 due to international character string requirements.

Table 254 provides a generic illustration of the encoding of a STRING value.

**Table 254 – STRING value**

|        | Contents<br>(charcount) |      | Contents<br>(string contents) |
|--------|-------------------------|------|-------------------------------|
| STRING | 0LSB                    | 1LSB | 0LSB                          |

1 octet characters

Table 255 provides a generic illustration of the encoding of a STRING2 value.

**Table 255 – STRING2 value**

|         | Contents<br>(charcount) |      | Contents<br>(string2contents) |      |
|---------|-------------------------|------|-------------------------------|------|
| STRING2 | 0LSB                    | 1LSB | 0LSB                          | 1LSB |

2 octet characters

Table 256 provides a generic illustration of the encoding of a STRINGN value.

**Table 256 – STRINGN value**

|         | Contents<br>(charsize) |      | Contents<br>(charcount) |      | Contents<br>(stringNcontents) |      |
|---------|------------------------|------|-------------------------|------|-------------------------------|------|
| STRINGN | 0LSB                   | 1LSB | 0LSB                    | 1LSB | 0LSB                          | NLSB |

N octet characters

Table 257 provides a generic illustration of the encoding of a SHORT\_STRING value.

**Table 257 – SHORT\_STRING value**

|              | Contents<br>(charcount) | Contents<br>(short_string) |
|--------------|-------------------------|----------------------------|
| SHORT_STRING | 0LSB                    | 0LSB                       |

1 octet characters

Table 258 illustrates the encoding of a string variable whose contents equal "Mill". Encoding examples of all string types are presented. Character coding is specified in ISO/IEC 10646. The hexadecimal equivalent is: 0x4D696C6C for 8-bit encoding.

Table 258 encodes "Mill" as a STRING type.

**Table 258 – Example compact encoding of a STRING value**

|        | Contents<br>(charcount) |    | Contents<br>(string contents) |    |    |    |
|--------|-------------------------|----|-------------------------------|----|----|----|
| STRING | 04                      | 00 | 4D                            | 69 | 6C | 6C |

Table 259 encodes "Mill" as a STRING2 type.

**Table 259 – Example compact encoding of STRING2 value**

|         | Contents<br>(charcount) |    | Contents<br>(string2 contents) |    |    |    |    |    |    |    |
|---------|-------------------------|----|--------------------------------|----|----|----|----|----|----|----|
| STRING2 | 04                      | 00 | 4D                             | 00 | 69 | 00 | 6C | 00 | 6C | 00 |

Table 260 encodes "Mill" as a SHORT\_STRING type.

**Table 260 – SHORT\_STRING type**

|              | Contents<br>(charcount) | Contents<br>(short_string contents) |    |    |    |
|--------------|-------------------------|-------------------------------------|----|----|----|
| SHORT_STRING | 04                      | 4D                                  | 69 | 6C | 6C |

### 5.1.3.7 FixedLengthBitString encoding

Subclause 5.1.3.7 provides examples of the Compact Encoding of SWORD, WORD, DWORD, LWORD data values. Figure 15 illustrates the bit placement rules associated with the Compact Encoding of a FixedLengthBitString.

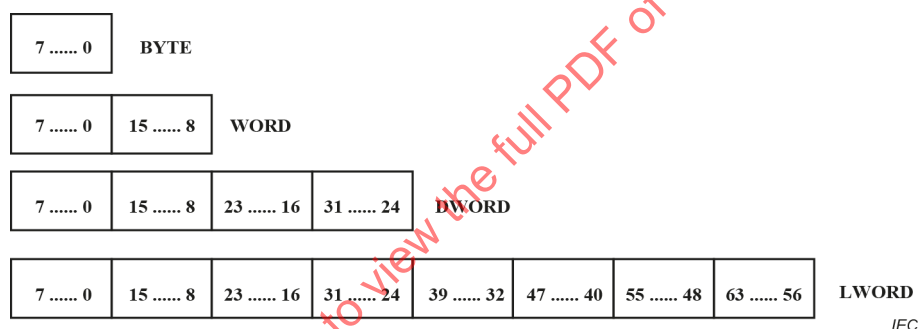
**Figure 15 – FixedLengthBitString compact encoding bit placement rules**

Figure 16 through Figure 19 illustrate the encoding of SWORD, WORD, DWORD, and LWORD.

Bits In Memory: 7 ..... 0  
00001111

Compact Encoded BYTE  
00001111 or 0x0F  
IEC

**Figure 16 – Example compact encoding of a SWORD FixedLengthBitString**

Bits In Memory: 15 ..... 0  
00001111 11001111

Compact Encoded WORD  
11001111 00001111 or 0xCF0F  
IEC

**Figure 17 – Example compact encoding of a WORD FixedLengthBitString**

Bits In Memory: 31 ..... 0  
00001111 11001111 10110110 00111110

Compact Encoded DWORD  
00111110 10110110 11001111 00001111 or 0x3EB6CF0F  
IEC

**Figure 18 – Example compact encoding of a DWORD FixedLengthBitString**

Bits In Memory: 63 ..... 0  
11110000 11001111 10110110 00111110 11110000 00101101 00011110 00001111

Compact Encoded LWORD  
00001111 00011110 00101101 11110000 00111110 10110110 11001111 11110000 or 0x0F1E2DF03EB6CF0  
IEC

**Figure 19 – Example compact encoding of a LWORD FixedLengthBitString**

### 5.1.3.8 Array encoding

#### 5.1.3.8.1 One-dimensional array encoding

The Array encoding shall use the encoding rules for the data types for each element and concatenates the elements which compose the array. The encoded values of the array elements shall be encoded in the same order as they are declared in the corresponding type or variable specification. These elements may be of any data type.

The definition of a single-dimensional array in the control network shall be:

ARRAY ::= SEQUENCE OF { array\_elements, NetworkData }

Assume the following array definition::

ARRAY1 ::= SEQUENCE OF {array\_elements:=2, UINT}

Plugging the UINT values {1,2} into this array definition yields the encoding specified in Table 261.

**Table 261 – Example compact encoding of a single dimensional ARRAY**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|
| ARRAY        | 01              | 00              | 02              | 00              |

#### 5.1.3.8.2 Two-dimensional array encoding

ARRAY ::= SEQUENCE OF { array\_elements  
SEQUENCE OF { array\_elements, NetworkData } }

#### 5.1.3.8.3 Three-dimensional array encoding

ARRAY ::= SEQUENCE OF { array\_elements,  
SEQUENCE OF { array\_elements,  
SEQUENCE OF { array\_elements,  
NetworkData } } }

Since control network data may comprise either ElementaryData or DerivedData, a new type or variable specification can be required before transmitting the values for the ARRAY.

A multi-dimensional array shall be seen on the wire as a single-dimensional array. The order of the array elements shall be maintained via the packing/unpacking sequence followed by the end nodes. The sequence followed shall be to access the Nth dimension of the array for all values of the other dimensions.

This shall be achieved by first accessing the array with all dimensions set to their initial index values. After this the Nth dimension is incremented through all of its index values. When the end of the index range for the Nth dimension is reached, the (N-1)th dimension is incremented, and the Nth dimension is set to its initial index value. This process is repeated until all of the array's dimensions have reached the ends of their index ranges, and results in the array being packed into the message buffer as a single-dimensional array. The same procedure is followed to unpack the single-dimensional array into a multi-dimensional array.

The two-dimensional array

ARRAY1 [2,3] of UINT:= { { 1, 2, 3 }, { 4, 5, 6 } }

results in the data stream shown in Table 262 when it is packed into a single-dimensional array following the compact encoding rules:

**Table 262 – Example compact encoding of a multi-dimensional ARRAY**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> | 5 <sup>th</sup> | 6 <sup>th</sup> | 7 <sup>th</sup> | 8 <sup>th</sup> | 9 <sup>th</sup> | 10 <sup>th</sup> | 11 <sup>th</sup> | 12 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|------------------|------------------|
| ARRAY        | 01              | 00              | 02              | 00              | 03              | 00              | 04              | 00              | 05              | 00               | 06               | 00               |

### 5.1.3.9 Structure encoding

The structure encoding shall use the encoding rules for the data types for each element and concatenates the elements which compose the structure.

The encoded values of the structure elements shall be encoded in the same order as they are declared in the corresponding ASN.1 type or variable specification. These elements may be of any Data type.

STRUCT::= SEQUENCE OF NetworkData -- May be different types

Since NetworkData may be comprised of either ElementaryData or DerivedData, a new type or variable specification can be required before transmitting the values for the STRUCT.

Assume the following structure definition:

newStruct::= SEQUENCE { BOOL,UINT,DINT}

Plugging the values {TRUE,0x1234,0x56789ABC} into the structure results in the encoding as specified in Table 263.

**Table 263 – Example compact encoding of a STRUCTURE**

| Octet number | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> | 5 <sup>th</sup> | 6 <sup>th</sup> | 7 <sup>th</sup> |
|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| newStruct    | 01              | 34              | 12              | BC              | 9A              | 78              | 56              |

### 5.1.3.10 Complex encoded data format examples

#### 5.1.3.10.1 General

The examples in 5.1.3.10.2 and 5.1.3.10.3 show how more complex data formats shall be packed. Example 5.1.3.10.2 shows the packing of an array of structures. Example in 5.1.3.10.3 shows how a structure with an array element is packed.

#### 5.1.3.10.2 Example 1: encoding an array of structures:

```
STRUCT1 ::= SEQUENCE OF {
    UINT      ele1;
    USINT     ele2;
    USINT     ele3;
    USINT     ele4;
    UINT      ele5;
}

ARRAY1 [ 2, 3 ] of STRUCT1 := {
    { { 1, 2, 3, 4, 5 }, { 6, 7, 8, 9, 10 },
      { 11, 12, 13, 14, 15 } },
    { { 15, 14, 13, 12, 11 }, { 10, 9, 8, 7, 6 },
      { 5, 4, 3, 2, 1 } } }
```

results in the following data stream:

```
[01][00][02][03][04][05][00] [06][00][07][08][09][0A][00]
[0B][00][0C][0D][0E][0F][00] [0F][00][0E][0D][0C][0B][00]
[0A][00][09][08][07][06][00] [05][00][04][03][02][01][00]
```

#### 5.1.3.10.3 Example 2: encoding a structure with an array element

```
STRUCT2 ::= SEQUENCE OF {
    UINT      ele1;
    ARRAY [ 3 ] of USINT array2;
    UINT      ele5;
}

STRUCT2 := { 1, { 2, 3, 4 }, 5 }
```

results in the following data stream:

```
[01][00] [02][03][04] [05][00]
```

## 5.2 Data type reporting

### 5.2.1 Object data representation

Objects may choose to implement a mechanism for reporting Data Type of a particular Attribute or transmitting type information along with the actual data. Subclause 5.2 defines the means by which Data Typing information is conveyed.

The specification of Data Type information utilizes the ASN.1 methodology specified in ISO/IEC 8824-1 and ISO/IEC 8825-1 with the control network defined optimizations to encode the **DataTypeSpecification** production defined in 4.2.4.

The control network defined optimization is that the Length Octet of a NULL type is not encoded.

NOTE For example, the encoding of 'abc [PRIVATE 1] IMPLICIT NULL' would be: 0xC1 (a tag with no length octet).

### 5.2.2 Elementary data type reporting

Elementary data types shall be identified using the identification codes defined in Table 264. These codes shall define the encoding of the primitive members of the **DataTypeSpecification** production. The control network specifies that ASN.1 NULL types shall not report the Length Octet of zero (0).

**Table 264 – Identification codes and descriptions of elementary data types**

| Data type name | Data type code (in hex) | Data type description                                             |
|----------------|-------------------------|-------------------------------------------------------------------|
| UTIME          | C0                      | System time information                                           |
| BOOL           | C1                      | Logical Boolean with values TRUE and FALSE                        |
| SINT           | C2                      | Signed 8-bit integer value                                        |
| INT            | C3                      | Signed 16-bit integer value                                       |
| DINT           | C4                      | Signed 32-bit integer value                                       |
| LINT           | C5                      | Signed 64-bit integer value                                       |
| USINT          | C6                      | Unsigned 8-bit integer value                                      |
| UINT           | C7                      | Unsigned 16-bit integer value                                     |
| UDINT          | C8                      | Unsigned 32-bit integer value                                     |
| ULINT          | C9                      | Unsigned 64-bit integer value                                     |
| REAL           | CA                      | 32-bit floating point value                                       |
| LREAL          | CB                      | 64-bit floating point value                                       |
| STIME          | CC                      | System time information                                           |
| DATE           | CD                      | Date information                                                  |
| TIME_OF_DAY    | CE                      | Time of day                                                       |
| DATE_AND_TIME  | CF                      | Date and time of day                                              |
| STRING         | D0                      | character string (1 octet per character)                          |
| SWORD          | D1                      | bit string - 8-bits                                               |
| WORD           | D2                      | bit string - 16-bits                                              |
| DWORD          | D3                      | bit string - 32-bits                                              |
| LWORD          | D4                      | bit string - 64-bits                                              |
| STRING2        | D5                      | character string (2 octets per character)                         |
| FTIME          | D6                      | Duration (high resolution)                                        |
| LTIME          | D7                      | Duration (long)                                                   |
| ITIME          | D8                      | Duration (short)                                                  |
| STRINGN        | D9                      | character string (N octets per character)                         |
| SHORT_STRING   | DA                      | character sting (1 octet per character, 1 octet length indicator) |
| TIME           | DB                      | Duration (milliseconds)                                           |
| EPATH          | DC                      | Path segments                                                     |
| STRINGI        | DE                      | International Character String                                    |
| NTIME          | DF                      | Duration (nanoseconds)                                            |
|                | E0 – E3                 | Reserved for future use                                           |

### 5.2.3 Constructed data type reporting

#### 5.2.3.1 General

Subclause 5.2.3 details the means by which the structure and array information presented within the DataTypeSpecification production is represented.

Table 265 specifies the identification codes used for the various constructed data types.

**Table 265 – Identification codes and descriptions of constructed data types**

| Data type name          | Data type code (in hex) | Data type description                                   |
|-------------------------|-------------------------|---------------------------------------------------------|
| Abbreviated Structure   | A0                      | Structure identified by 16-bit CRC value                |
| Abbreviated Array       | A1                      | Array identified by 16-bit CRC value                    |
| Formal Structure        | A2                      | Structure defined by member type list                   |
| Formal Array            | A3                      | Array defined by type and size                          |
|                         | A4-A7                   | Reserved – do not use                                   |
| Formal Handle Structure | A8                      | Structure defined by member type and member handle list |
|                         | A9-BF                   | Reserved                                                |

### 5.2.3.2 Structure type definition

#### 5.2.3.2.1 General

The control network defines three different methods for reporting Structure type definitions:

- Formal Encoding (FormalStrucTypeSpec);
- Formal Handle Encoding (FormalHandleStrucTypeSpec);
- Abbreviated Encoding (AbbreviatedStrucTypeSpec).

Formal encoding shall be used to provide a detailed report of the complete structure definition, including the complete definition of all component data types (the structure member number can be obtained based on the component position). Formal Handle encoding is used to provide a detailed report of the complete structure definition, including the complete definition of all component data types and component handles. Abbreviated encoding shall be used to specify a shorter form of the structure definition. This shorter form shall not include the data types associated with the structure's components.

#### 5.2.3.2.2 Formal encoding for structure type information

Table 266 defines the formal encoding format for non-nested structures.

**Table 266 – Formal structure encoding definition**

| Octet      | Definition                                   |
|------------|----------------------------------------------|
| 0          | Formal Structure (A2)                        |
| 1          | Length of structure definition in octets (n) |
| 2 to (n+1) | Data type code (C0 to DF)                    |

The examples in 5.2.3.2.3 and 5.2.3.2.4 illustrate formal encoding for structure type specifications. This is actually an example of the encoding of the FormalStrucTypeSpec production defined in 4.2.4.

#### 5.2.3.2.3 Example 1

Figure 20 illustrates the encoding of the following structure definition.

```
STRUCT ::= SEQUENCE OF { BOOL, UINT, DINT }
```

| STRUCT type | Type length | Data type (BOOL) | Data type (UINT) | Data type (DINT) |
|-------------|-------------|------------------|------------------|------------------|
| A2          | 03          | C1               | C7               | C4               |

NOTE The IMPLICIT NULL types from the DataTypeSpecification production are not followed by a Length Octet of zero (0).

**Figure 20 – Example 1 of formal encoding of a structure type specification****5.2.3.2.4 Example 2**

Figure 21 illustrates the encoding of the following structure definition.

```
STRUCT_MAIN ::= SEQUENCE OF { UINT, STRUCT_SUB, INT }
```

with subelement STRUCT\_SUB defined as:

```
STRUCT_SUB ::= SEQUENCE OF { UINT, SINT, INT }
```

| Struct type | Type length | Component        |             |             |                  |                  |                 |                 |
|-------------|-------------|------------------|-------------|-------------|------------------|------------------|-----------------|-----------------|
|             |             | Data type (UINT) | Struct type | Type length | Nested Component |                  |                 | Data type (INT) |
|             |             |                  |             |             | Data type (UINT) | Data type (SINT) | Data type (INT) |                 |
| A2          | 07          | C7               | A2          | 03          | C7               | C2               | C3              | C3              |

**Figure 21 – Example 2 of formal encoding of a structure type specification****5.2.3.2.5 Formal Encoding for Handle Structure Type Information**

Table 267 defines the formal encoding format for structures with handles.

**Table 267 – Formal structure with handles encoding definition**

| Octet                        | Definition                                                          |
|------------------------------|---------------------------------------------------------------------|
| 0                            | Formal Handle Structure (A8)                                        |
| 1                            | Length of structure definition in octets (n)                        |
| 2                            | Handle length (1, 2 or 4 octets)                                    |
| 3                            | Data type code (C0 to DF) 1                                         |
| 4 to 4 + (Handle length – 1) | Handle (1, 2 or 4 octet value assigned by application) <sup>a</sup> |

<sup>a</sup> Additional members are represented by pairs of data type codes and handles.

The two examples in 5.2.3.2.6 and 5.2.3.2.7 illustrate formal encoding for handle structure type specifications. This is actually an example of the encoding of the FormalHandleStrucTypeSpec production defined in 4.2.4.

**5.2.3.2.6 Example 3**

Figure 22 shows the encoding of the following structure definition.

```
STRUCT ::= SEQUENCE OF { BOOL, UINT, DINT }
```

| Struc type | Type length | Handle length | Data type (BOOL) | Member handle | Data type (UINT) | Member handle | Data type (DINT) | Member handle |
|------------|-------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|
| A8         | 10          | 04            | C1               | D9 B9 74 3E   | C7               | F4 15 59 93   | C4               | 0B 7B 24 A6   |

**Figure 22 – Example 3 of formal encoding of a handle structure type specification**

#### 5.2.3.2.7 Example 4

Figure 23 shows the encoding of the following structure definition

```
STRUCT_MAIN ::= SEQUENCE OF { UINT, STRUCT_SUB, INT }
```

with sub element STRUCT\_SUB defined as:

```
STRUCT_SUB ::= SEQUENCE OF { SINT, INT }
```

| Struc type | Type length | Handle length | Component        |               |            |             |               |                  |               |                 |               |                 |        |
|------------|-------------|---------------|------------------|---------------|------------|-------------|---------------|------------------|---------------|-----------------|---------------|-----------------|--------|
|            |             |               | Data type (UINT) | Member handle | Struc type | Type length | Handle length | Nested component |               |                 |               | Data Type (INT) | Handle |
|            |             |               |                  |               |            |             |               | Data Type (SINT) | Member Handle | Data Type (INT) | Member Handle |                 |        |
| A8         | 0E          | 02            | C7               | 90 1D         | A8         | 05          | 01            | C2               | F8            | C3              | B2            | C3              | 36 ED  |

**Figure 23 – Example 4 of formal encoding of a handle structure type specification**

#### 5.2.3.2.8 Abbreviated encoding for structure type information

Table 268 defines the abbreviated encoding format for structures.

**Table 268 – Abbreviated structure encoding definition**

| Octet  | Definition                           |
|--------|--------------------------------------|
| 0      | Abbreviated Structure (A0)           |
| 1      | Length of abbreviated structure (02) |
| 2 to 3 | UINT containing CRC                  |

The example in 5.2.3.2.9 illustrates the abbreviated encoding for structure type specifications. This is actually an example of the encoding of the AbbreviatedStrucTypeSpec production defined in 4.2.4.

The UINT defined within the AbbreviatedStrucTypeSpec production shall be initialized with a 16 bit Cyclic Redundancy Check (CRC) value (see IEC 61158-4-2). This can be used by application logic to determine whether or not the format of the structure has changed. The octet stream used to produce the CRC is the actual formally encoded (FormalStrucTypeSpec) structure type specification.

#### 5.2.3.2.9 Example 5

Figure 24 shows the abbreviated encoding of the structure definition presented in 5.2.3.2.8:

| Struct Type | Type Length | UINT containing CRC |    |
|-------------|-------------|---------------------|----|
| A0          | 02          | C7                  | 26 |

IEC

**Figure 24 – Example 5 of abbreviated encoding of a structure type specification**

NOTE The algorithms presented in this document are used to generate the CRC value of 0x26C7 from the Formally Encoded type specification: [A2][07][C7][A2][03] [C7][C2][C3][C3].

### 5.2.3.3 Array type definition

#### 5.2.3.3.1 General

Two different methods for reporting array type definitions shall be:

- Formal Encoding (FormalArrayTypeSpec);
- Abbreviated Encoding (AbbreviatedArrayTypeSpec).

Formal encoding shall be used to provide a detailed report of the complete array definition, including the data content and the array's dimensions. Abbreviated encoding shall be used to specify a shorter form of the array definition. This shorter form shall not include information specifying the array's dimensions.

#### 5.2.3.3.2 Formal encoding for array type information

Table 269 defines the formal encoding format for arrays.

**Table 269 – Formal array encoding definition**

| Octet                                                                           | Definition                                                                                                                                                        |
|---------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0                                                                               | Formal Array (A3)                                                                                                                                                 |
| 1 to 4                                                                          | Number of elements in the dimension (UDINT)                                                                                                                       |
| 5 to end                                                                        | One of:<br>1 Array element data type code (C0 to DF)<br>2 Formal array encoding <sup>a</sup><br>3 Formal structure encoding<br>4 Formal handle structure encoding |
| <sup>a</sup> Each dimension of the array shall be represented by a formal array |                                                                                                                                                                   |

The examples in 5.2.3.3.3 and 5.2.3.3.4 illustrate formal encoding for structure type specifications. This is actually an example of the encoding of the FormalArrayTypeSpec production defined in 4.2.4.

#### 5.2.3.3.3 Example 1

Figure 25 shows the formal encoding of the following array definition (one dimensional array of 10 UINT elements).

```
ARRAY1 ::= SEQUENCE OF { array_elements := 10, UINT }
```

| Array | Array elements (octet 0) | Array elements (octet 1) | Array elements (octet 2) | Array elements (octet 3) | Data type (UINT) |
|-------|--------------------------|--------------------------|--------------------------|--------------------------|------------------|
| A3    | 0A                       | 800                      | 000                      | 000                      | C7               |

**Figure 25 – Example 1 of formal encoding of an array type specification**

NOTE The IMPLICIT NULL types from the DataTypeSpecification production are not followed by a Length Octet of zero (0).

#### 5.2.3.3.4 Example 2

Figure 26 shows the encoding of the following array definition (two dimensional array of 20 elements by 256 elements of a structure consisting of a UINT, SINT and INT).

```
ARRAY1 ::= SEQUENCE OF { array_elements:= 20,
                        SEQUENCE OF { array_elements:= 256, STRUCT_ELEMENT } }
```

in which STRUCT\_ELEMENT is defined as:

```
STRUCT_ELEMENT ::= SEQUENCE OF { UINT, SINT, INT }
```

Formal Encoding: [A3][14][00][00][00][A3][00][01][00][00][A2][03][C7][C2][C3]

| Array type | Array elements (octet 0) | Array elements (octet 1) | Array elements (octet 2) | Array elements (octet 3) |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|
| A3         | 14                       | 00                       | 00                       | 00                       |

| Array contents |                          |                          |                          |                          |
|----------------|--------------------------|--------------------------|--------------------------|--------------------------|
| Array Type     | Array elements (octet 0) | Array elements (octet 1) | Array elements (octet 2) | Array elements (octet 3) |
| A3             | 00                       | 01                       | 00                       | 00                       |

| Nested array contents |             |      |      |     |
|-----------------------|-------------|------|------|-----|
| Struct type           | Type length | UINT | SINT | INT |
| A2                    | 03          | C7   | C2   | C3  |

Figure 26 – Example 2 of formal encoding of an array type specification

NOTE The IMPLICIT NULL types from the DataTypeSpecification production are not followed by a Length Octet of zero (0).

#### 5.2.3.3.5 Abbreviated tag encoding for array type information

Table 270 defines the abbreviated encoding format for arrays.

Table 270 – Abbreviated array encoding definition

| Octet                                                                                  | Definition                                                                                                                            |
|----------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| 0                                                                                      | Abbreviated Array (A1)                                                                                                                |
| 1                                                                                      | Length of abbreviated array (n)                                                                                                       |
| 2 to (2+n-1)                                                                           | One of:<br>1 Array element data type code (C0 to DF)<br>2 Abbreviated array encoding <sup>a</sup><br>3 Abbreviated structure encoding |
| <sup>a</sup> Each dimension of the array shall be represented by an abbreviated array. |                                                                                                                                       |

The abbreviated encoding of an Array type shall not include the information specifying the Array's dimensions. This is actually an example of the encoding of the AbbreviatedArrayTypeSpec production defined in 4.2.4.

### 5.2.3.3.6 Example 1

Figure 27 shows the abbreviated encoding of the following array definition (one dimensional array of 10 UINT elements):

```
ARRAY2 ::= SEQUENCE OF { array_elements := 10, UINT }
```

| Array type | Type length | Data type (UINT) |
|------------|-------------|------------------|
| A1         | 01          | C7               |

**Figure 27 – Example 1 of abbreviated encoding of an array type specification**

NOTE The IMPLICIT NULL types from the DataTypeSpecification production are not followed by a Length Octet of zero (0).

### 5.2.3.3.7 Example 2

Figure 28 shows the abbreviated encoding of the following array definition (two dimensional array of 20 elements by 900 elements of a structure consisting of a UINT, SINT and INT):

```
ARRAY ::= SEQUENCE OF { array_elements := 20,
                        SEQUENCE OF { array_elements := 900, STRUCT_ELE } }
```

in which STRUCT\_ELE is defined as:

```
STRUCT_ELE ::= SEQUENCE OF { UINT, SINT, INT }
```

| Array type | Type length | Array type | Type length | Structure (STRUCT_ELE) |             |                     |    |
|------------|-------------|------------|-------------|------------------------|-------------|---------------------|----|
|            |             |            |             | Struct type            | Type length | UINT containing CRC |    |
| A1         | 06          | A1         | 04          | A0                     | 02          | 59                  | 51 |

**Figure 28 – Example 2 of abbreviated encoding of an array type specification**

NOTE The algorithms presented in this document are used to generate the CRC value of 0x5159 from the Formally Encoded type specification: [A2][03][C7][C2][C3].

## 6 Structure of FAL protocol state machines

Interface to FAL services and protocol machines are specified in Clause 6. Conventions used for the descriptions are given in IEC 61158-1.

This fieldbus follows the structure outlined for Type 1 fieldbus with the following specific features:

- There is no formal definition of AP-Context Machine
- There is a formally-defined FSPM Machine serving as an interface between FAL User and ARPM.
- There are ARPM Machines of two different types:
  - 1) one ARPM machine for connection-less application relationships
  - 2) seven ARPM machines for connection-oriented application relationships
- DMPM Machines are defined at the interface to the supported Data-link layer (Type 2 or others).

## 7 AP-Context state machine

### 7.1 Overview

Type 2 supports the AP-Context State Machine specified in 7.2, when the Connection object is also supported.

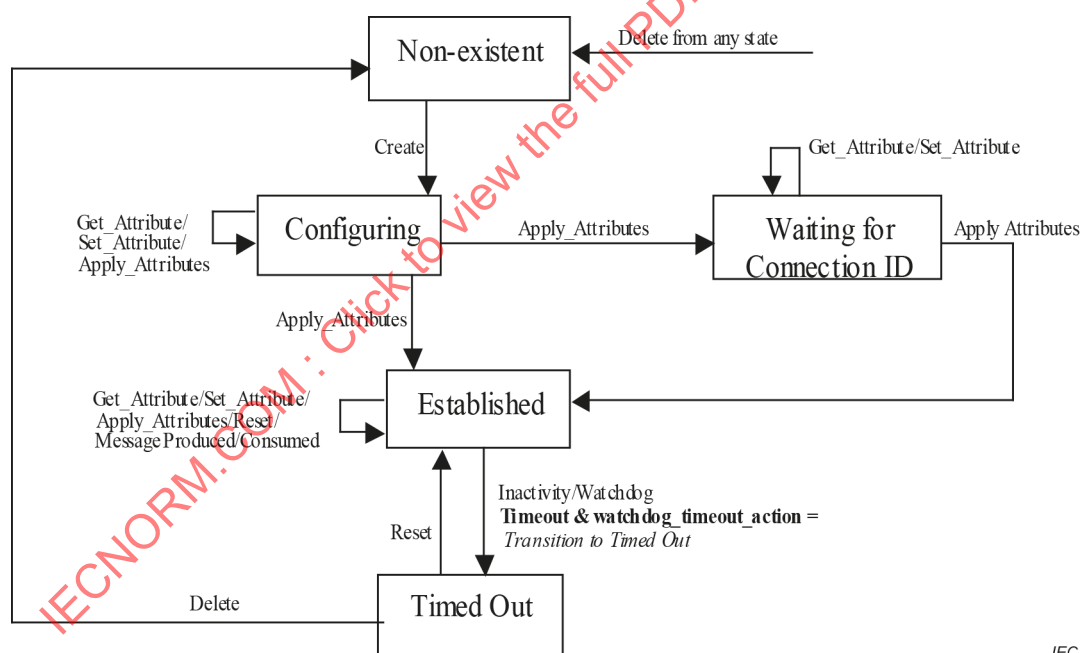
These state machines refer to the following additional internal services.

- **Send\_Message** – Used internally to trigger the transmission of a message. This results in the invocation of the associated Link Producer's Send service. If the message needs to be transmitted in a fragmented fashion, this service breaks up the message into multiple fragments and invokes the associated Link Producer's Send service multiple times.
- **Receive\_Data** – Used internally to deliver a received frame to the Connection object. This service is invoked internally by a Link Consumer. The Connection object is responsible for determining whether or not it shall reassemble a series of data fragments into a complete message before processing the message.

### 7.2 Connection object state machine

#### 7.2.1 I/O Connection instance behavior

Figure 29 provides a general overview of the behavior associated with an **I/O Connection object** (**instance\_type** attribute = I/O).



IEC

**Figure 29 – I/O Connection object state transition diagram**

**Important:** Table 271 provides a detailed State Event Matrix for an I/O Connection object and implementations should be based on this information. This State Event Matrix does not dictate rules with regards to product specific, internal logic. Any attempt to access the Connection Class or a Connection object Instance can need to pass through product specific verification. This can result in an error scenario that is not indicated by the SEM in Table 271. This can also result in additional, product specific indications delivered from a Connection object to the application and/or a specific application object. The point to remember is that the Connection object shall exhibit the externally visible behavior specified by the SEM and the attribute definitions.

**Table 271 – I/O Connection state event matrix**

| Event                                             | I/O Connection object state                                                                                                             |                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                               |                                                                                                               |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
|                                                   | Non-Existent                                                                                                                            | Configuring                                                                                       | Waiting for Connection ID                                                                                                                                                                                                                                                                                                                                                               | Established                                                                                                   | Timed Out                                                                                                     |
| <b>Connection Class receives a Create Request</b> | Class instantiates a Connection object. Set instance_type to I/O. Set all other attributes to default values. Transition to Configuring | Not applicable                                                                                    | Not applicable                                                                                                                                                                                                                                                                                                                                                                          | Not applicable                                                                                                | Not applicable                                                                                                |
| <b>Connection Class receives a Delete Request</b> | Error: Object does not exist (General Error Code 0x16)                                                                                  | Release all associated resources. Transition to Non-existent                                      | Release all associated resources. Transition to Non-existent.                                                                                                                                                                                                                                                                                                                           | Release all associated resources. Transition to Non-existent.                                                 | Release all associated resources. Transition to Non-existent.                                                 |
| <b>Set_Attribute_Single</b>                       | Error: Object does not exist (General Error Code 0x16)                                                                                  | Validate/service the request based on internal logic and per the Access Rules<br>Return response. | If request to modify produced or consumed_connection_id then validate the value and service the request. Return appropriate response. If this is a request to access an attribute other than the produced or consumed_connection_id, then return an Error Response whose General Error Code is set to 0x0C (The object can not perform the requested service in its current mode/state) | Validate/service the request based on internal logic and per the Access Rules<br>Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules<br>Return appropriate response. |
| <b>Get_Attribute_Single</b>                       | Error: Object does not exist (General Error Code 0x16)                                                                                  | Validate/service the request based on internal logic and per the Access Rules<br>Return response. | Validate/service the request based on internal logic and per the Access Rules<br>Return response.                                                                                                                                                                                                                                                                                       | Validate/service the request based on internal logic and per the Access Rules<br>Return response.             | Validate/service the request based on internal logic and per the Access Rules<br>Return response.             |

| Event        | I/O Connection object state                            |                                                                                                                     |                                                                                                                     |                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                           |
|--------------|--------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | Non-Existent                                           | Configuring                                                                                                         | Waiting for Connection ID                                                                                           | Established                                                                                                                                                                                                                                                                                                        | Timed Out                                                                                                                                                                                                                                                                                                                                                 |
| <b>Reset</b> | Error: Object does not exist (General Error Code 0x16) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0x0C) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0x0C) | Cancel the current Inactivity/Watchdog Timer. Using the value in the expected_packet_rate attribute, re-start the Inactivity/Watchdog Timer. A success response is returned even if an Inactivity/Watchdog Timer is not utilized by the Connection object (Client Transport Class 0, expected_packet_rate = 0x0C). | Using the value in the expected_packet_rate attribute, start the Inactivity/Watchdog timer and transition back to the Established state.<br><br>If the expected_packet_rate attribute has been set to zero (0) while the Connection was in the Timed Out state, then just transition back to Established without activating an Inactivity/Watchdog Timer. |

IECNORM.COM : Click to view the full PDF of IEC 61158-6-2:2023

| Event                   | I/O Connection object state                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                    |                                                                                                                     |
|-------------------------|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
|                         | Non-Existent                                           | Configuring                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Waiting for Connection ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Established                                                                                                                                                                                                                        | Timed Out                                                                                                           |
| <b>Apply_Attributes</b> | Error: Object does not exist (General Error Code 0x16) | Deliver Connection object to the application which validates the attribute information. If either of the connection_id attributes (produced or consumed) needs to be configured and cannot be generated by this module, then perform all the other steps necessary to configure the connection, return a Successful Response and transition to the Waiting For Connection ID state. The inability to generate a produced or consumed connection ID value IS NOT reported as an error. If all attributes are validly configured, then perform all the steps necessary to satisfy this connection, start all required timers, and transition to the Established state. If an error is detected, then an Error Response is returned and the Connection remains in the Configuring state <sup>a</sup> and, if a Client Connection with a production trigger value of 0 or 1 (Cyclic or Change of State), produce initial data. | If either of the connection_id attributes (produced or consumed) still needs to be configured and cannot be generated by this module, then return a Successful Response and remain in the Waiting For Connection ID state. The inability to generate a produced or consumed connection ID value IS NOT reported as an error. If the produced and/or consumed_connection_id attributes are now validly configured, then transition to the Established state and return a Successful Response <sup>a</sup> and, if a Client Connection with a production trigger value of 0 or 1 (Cyclic or Change of State), produce initial data. | All modifications take place immediately once the Connection has transitioned to the Established state. Return Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0x0C) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0x0C) |

| Event                                    | I/O Connection object state |                                                 |                                                 |                                                                                                                                                                                                                                                                                                                                                                                                        |                                                 |
|------------------------------------------|-----------------------------|-------------------------------------------------|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
|                                          | Non-Existent                | Configuring                                     | Waiting for Connection ID                       | Established                                                                                                                                                                                                                                                                                                                                                                                            | Timed Out                                       |
| <b>Receive_Data</b>                      | Not applicable              | Discard the message                             | Discard the message                             | <p>If a complete, valid<sup>b</sup> message has been received, reset the Inactivity/Watchdog Timer<sup>c</sup> and deliver the I/O Message to the Application. A Connection object shall exhibit the externally visible behavior associated with the current state of its attributes (see Access Rules).</p> <p>If this is a fragmented portion of an I/O Message, process as specified by subnet.</p> | Discard the message                             |
| <b>Send_Message</b>                      | Not applicable              | Return internal error - do not send the message | Return internal error - do not send the message | Transmit the complete I/O Message or fragment as required by the subnet. If this is a Client Connection and the expected_packet_rate attribute is non-zero, restart the Transmission Trigger Timer.                                                                                                                                                                                                    | Return internal error - do not send the message |
| <b>Inactivity/Watchdog Timer expires</b> | Not applicable              | Not applicable                                  | Not applicable                                  | Examine the watchdog_timeout_action attribute of the Connection and perform the indicated action. If the watchdog_timeout_action attribute indicates that the Connection is to remain in the Established state ( <i>Auto Reset</i> ), then immediately re-start the Inactivity/Watchdog Timer.                                                                                                         | Not applicable                                  |

NOTE Attribute Access Rules are specified in IEC 61158-5-2, 6.2.3.2.1.4.

- <sup>a</sup> If the configuration indicates that a Message ID needs to be allocated and an available Message ID does not exist in the specified Message Group, then an Error Response whose General Error Code indicates *Resource Unavailable* (0x02) is returned. If a Connection object attribute value passed the range check when it was initially configured but the attribute value conflicts with another piece of information in the node when the Apply request is processed, then an Error Response is returned whose General Error Code is set to *Invalid Attribute Value* (0x09) and whose Extended Code is set to the Attribute ID of the *offending* Connection object Attribute ID.
- <sup>b</sup> The Connection object verifies that the length of the received I/O Message is less than or equal to the consumed\_connection\_size attribute prior to processing the message. If the length of the received message is less than or equal to the consumed\_connection\_size attribute, then the I/O Connection object resets the Inactivity/Watchdog Timer, exhibits the externally visible behavior indicated by its attribute settings, and delivers the message to the Application. If the length of the received message is greater than the consumed\_connection\_size attribute, then the I/O Connection object immediately discards the message and discontinues any subsequent processing. This is the only message content validation performed by an I/O Connection object. Subsequent validation shall be performed by the Application.
- <sup>c</sup> If a fragmented message is being received, then the Inactivity/Watchdog Timer is not reset until the entire message has been validly received.

**Important:** The Receive\_Data event is only delivered to an I/O Connection when a message whose Connection Identifier Field matches the consumed\_connection\_id attribute is received. If a message is received whose Connection Identifier Field does not match any Established Connection object's consumed\_connection\_id attribute, then the message is discarded. Connection Identifiers are subnet-type specific.

If an implementation detects that it does not support an Explicit Messaging Service indicated in Table 271, then an Error Response specifying a General Error Code 0x08 (Service Not Supported) is returned.

## 7.2.2 Bridged Connection instance behavior

Figure 30 provides a general overview of the behavior associated with a **Bridged** Connection object (**instance\_type** attribute = Bridged). Bridged connections are used to make connections offlink. Both I/O and Explicit Messaging can be accomplished using this connection type. The Connection Manager object definition provides more details these types of connections.

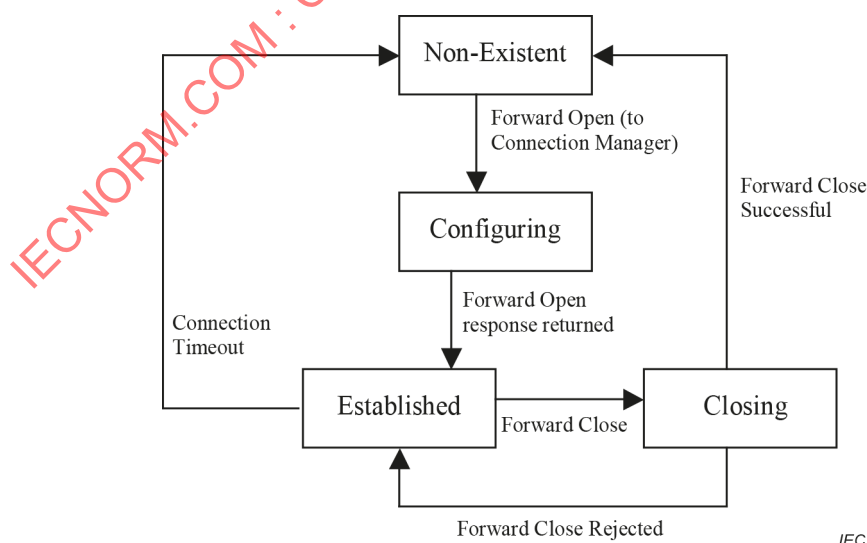


**Figure 30 – Bridged Connection object state transition diagram**

Table 272 provides a detailed State Event Matrix for a Bridged Connection object.

**Table 272 – Bridged Connection state event matrix**

| Event                                                                                        | Bridged Connection object state                                                                                                                                                                                                  |                                                                                                             |                                                                                                             |                                                                                                             |
|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
|                                                                                              | Non-Existent                                                                                                                                                                                                                     | Configuring                                                                                                 | Established                                                                                                 | Closing                                                                                                     |
| Connection Manager receives a Forward Open Request                                           | Connection Class instantiates a Connection object. Set <b>instance_type</b> to <b>Bridged</b> . Set attributes to value delivered by Forward Open service or default for Bridged connections. Transition to <b>Configuring</b> . | Not applicable                                                                                              | Not applicable                                                                                              | Not applicable                                                                                              |
| Connection Manager receives notification that connection establishment is complete to target | Not applicable                                                                                                                                                                                                                   | Transition to <b>Established</b>                                                                            | Not applicable                                                                                              | Not applicable                                                                                              |
| Connection Manager receives a Forward Close Request                                          | Not applicable                                                                                                                                                                                                                   | Ignore event                                                                                                | Transition to <b>Closing</b>                                                                                | Ignore event                                                                                                |
| Connection Manager receives a Forward Close Response                                         | Not applicable                                                                                                                                                                                                                   | Release all resources and transition to <b>Non-Existent</b>                                                 | Release all resources and transition to <b>Non-Existent</b>                                                 | Release all resources and transition to <b>Non-Existent</b>                                                 |
| Delete                                                                                       | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                           | Release all resources and transition to <b>Non-Existent</b>                                                 | Release all resources and transition to <b>Non-Existent</b>                                                 | Release all resources and transition to <b>Non-Existent</b>                                                 |
| Get_Attribute_Single                                                                         | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                           | Validate/service the request based on internal logic and per the Access Rules. Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules. Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules. Return appropriate response. |
| Set_Attribute_Single                                                                         | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                           |                                                                                                             |                                                                                                             |                                                                                                             |
| Reset                                                                                        | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                           | Error: Service Not Supported (General Error Code 0x08)                                                      | Error: Service Not Supported (General Error Code 0x08)                                                      | Error: Service Not Supported (General Error Code 0x08)                                                      |
| Apply_Attributes                                                                             | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                           | Error: Service Not Supported (General Error Code 0x08)                                                      | Error: Service Not Supported (General Error Code 0x08)                                                      | Error: Service Not Supported (General Error Code 0x08)                                                      |
| Receive_Data                                                                                 | Not applicable                                                                                                                                                                                                                   | Ignore event                                                                                                | Invoke send service of Connection object on destination port, passing the received data.                    | Ignore event                                                                                                |
| Send_Message                                                                                 | Not applicable                                                                                                                                                                                                                   | Ignore event                                                                                                | Send data on subnet.                                                                                        | Ignore event                                                                                                |
| Inactivity/Watchdog Timer expires                                                            | Not applicable                                                                                                                                                                                                                   | Not applicable                                                                                              | Release all resources and transition to <b>Non-Existent</b>                                                 | Release all resources and transition to <b>Non-Existent</b>                                                 |
| NOTE Attribute Access Rules are specified in IEC 61158-5-2, 6.2.3.2.1.4.                     |                                                                                                                                                                                                                                  |                                                                                                             |                                                                                                             |                                                                                                             |

### 7.2.3 Explicit Messaging Connection instance behavior

Figure 31 provides a general overview of the behavior associated with an **Explicit Messaging Connection object** (**instance\_type** attribute = Explicit Messaging).

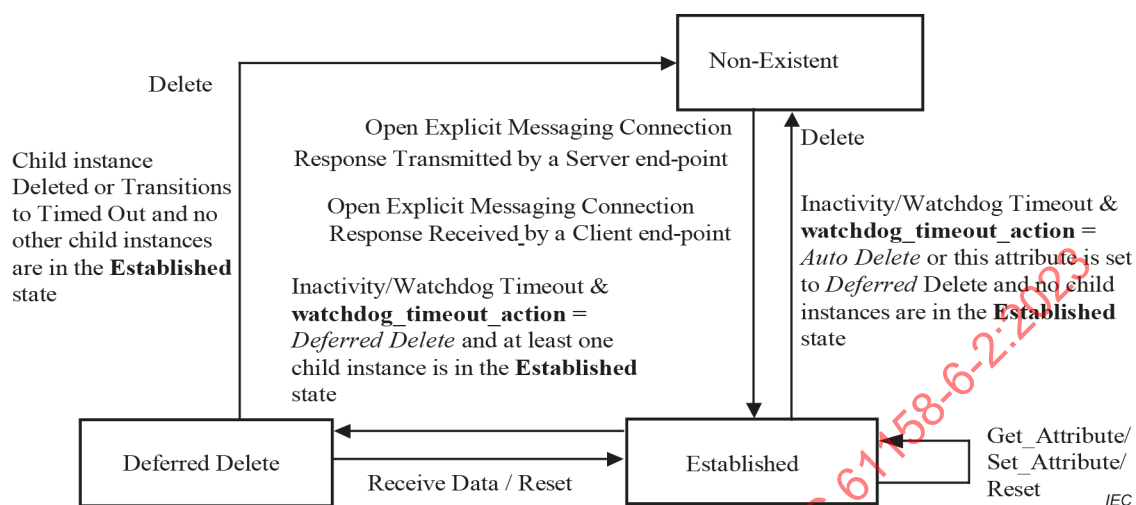


**Figure 31 – Explicit Messaging Connection object state transition diagram**

Table 273 provides a detailed State Event Matrix for an Explicit Messaging Connection object. Implementations should be based on the information in Table 273.

**Table 273 – Explicit Messaging Connection state event matrix**

| Event                                                                                                                       | Explicit Messaging Connection object state                                                                                                                                                                                                                                                                                                                                     |                                                                                                                |                                                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
|                                                                                                                             | Non-Existent                                                                                                                                                                                                                                                                                                                                                                   | Established                                                                                                    | Deferred Delete                                                                                                |
| UCMM receives an Open Explicit Messaging Connection Request/Response and invokes the Create service of the Connection Class | If possible, Class instantiates Connection object. Set <b>instance_type</b> attribute to <i>Explicit Messaging<sup>a</sup></i> . Other attributes are automatically configured using the information in the Open Explicit Messaging Connection Request/Response. Transition to <b>Established</b> . If request received, Transmit Open Explicit Messaging Connection Response. | Not applicable                                                                                                 | Not applicable                                                                                                 |
| UCMM receives a Close Request and invokes the Delete service of the Connection Class                                        | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                                                                                                                                                                         | Release all associated resources. Transition to <b>Non-existent</b> .                                          | Release all associated resources. Transition to <b>Non-existent</b> .                                          |
| Connection Class receives a Delete Request                                                                                  | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                                                                                                                                                                         | Release all associated resources. Transition to <b>Non-existent</b> .                                          | Release all associated resources. Transition to <b>Non-existent</b> .                                          |
| Set_Attribute_Single                                                                                                        | Error: Object does not exist (General Error Code 0x16)                                                                                                                                                                                                                                                                                                                         | Validate/service the request based on internal logic and per the Access Rules.<br>Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules.<br>Return appropriate response. |

| Event                                                                                                                                                                                                                                     | Explicit Messaging Connection object state             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                                                           | Non-Existent                                           | Established                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Deferred Delete                                                                                                                                                                                                             |
| Get_Attribute_Single                                                                                                                                                                                                                      | Error: Object does not exist (General Error Code 0x16) | Validate/service the request based on internal logic and per the Access Rules.<br><br>Return appropriate response.                                                                                                                                                                                                                                                                                                                                                    | Validate/service the request based on internal logic and per the Access Rules.<br><br>Return appropriate response.                                                                                                          |
| Reset                                                                                                                                                                                                                                     | Error: Object does not exist (General Error Code 0x16) | Cancel the current Inactivity/Watchdog Timer. Using the value in the <b>expected_packet_rate</b> attribute, re-start the Inactivity/Watchdog Timer.                                                                                                                                                                                                                                                                                                                   | Using the value in the <b>expected_packet_rate</b> attribute, re-start the Inactivity/Watchdog Timer and transition back to the Established state.                                                                          |
| Apply_Attributes                                                                                                                                                                                                                          | Error: Object does not exist (General Error Code 0x16) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0x0C)                                                                                                                                                                                                                                                                                                                                                   | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0x0C)                                                                                                         |
| Receive_Data                                                                                                                                                                                                                              | Not applicable                                         | If a valid message or message fragment has been received, then reset the Inactivity/Watchdog Timer. Either process/store the fragment or handle the Explicit Message.                                                                                                                                                                                                                                                                                                 | If a valid message or message fragment has been received, then restart the Inactivity/Watchdog Timer and transition back to the <b>Established</b> state. Either process/store the fragment or handle the Explicit Message. |
| Send_Message                                                                                                                                                                                                                              | Not applicable                                         | Examine the length of the message to transmit and, if necessary, perform a fragmented series of transmissions. Otherwise, transmit the complete Explicit Message.                                                                                                                                                                                                                                                                                                     | Examine the length of the message to transmit and, if necessary, perform a fragmented series of transmissions. Otherwise, transmit the complete Explicit Message.                                                           |
| Inactivity/Watchdog Timer expires                                                                                                                                                                                                         | Not applicable                                         | If the <b>watchdog_timeout_action</b> attribute is set to <i>Auto Delete</i> or is set to <i>Deferred Delete</i> and no child connection instances are in the <b>Established</b> state release all associated resources and Transition to <b>Non-existent</b> . If the <b>watchdog_timeout_action</b> attribute is set to <i>Deferred Delete</i> and at least one child connection instance is in the <b>Established</b> state transition to <b>Deferred Delete</b> . | Not applicable.                                                                                                                                                                                                             |
| Child connection instance Deleted or Transitions to Timed Out                                                                                                                                                                             | Not applicable                                         | Ignore event                                                                                                                                                                                                                                                                                                                                                                                                                                                          | If no other child connection instances are in the <b>Established</b> state release all associated resources and transition to <b>Non-existent</b> .                                                                         |
| NOTE Attribute Access Rules are specified in IEC 61158-5-2, 6.2.3.2.1.4.                                                                                                                                                                  |                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                             |
| <sup>a</sup> If the configuration indicates that a connection resource needs to be allocated and an available resource does not exist, then an Error Response whose General Error Code indicates Resource Unavailable (0x02) is returned. |                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                             |

**Important:** The Receive\_Data event is only delivered to an Explicit Messaging Connection when a message who's Connection ID matches the consumed\_connection\_id attribute is received. If a message is received whose connection id does not match any Established Connection object's consumed\_connection\_id attribute, then the message is discarded.

If an implementation detects that it does not support an Explicit Messaging Service indicated in Table 273, then an Error Response specifying a General Status Code 0x08 (Service Not Supported) is returned.

## 8 FAL service protocol machine (FSPM)

### 8.1 General

This fieldbus FAL Service Protocol State Machine maps FAL User services onto services of communication objects internal to FAL.

### 8.2 Primitive definitions

The Objects within FSPM shall provide the following services:

|                           |                                                                                                       |
|---------------------------|-------------------------------------------------------------------------------------------------------|
| Get_Attributes_All        | Reads values of all attributes of the specified object class or instance                              |
| Set_Attributes_All        | Writes specified values to all attributes of the specified object class or instance                   |
| Get_Attribute_List        | Reads values of the specified list of attributes of the specified object class or instance            |
| Set_Attribute_List        | Writes specified values to the specified list of attributes of the specified object class or instance |
| Get_Attribute_Single      | Reads value of the specified attribute of the specified object class or instance                      |
| Set_Attribute_Single      | Writes specified value to the specified attribute of the specified object class or instance           |
| Reset                     | Resets the specified object class or instance                                                         |
| Create                    | Creates an instance of the specified object class                                                     |
| Delete                    | Deletes an instance of the specified object class                                                     |
| Start                     | Starts execution of the specified object                                                              |
| Stop                      | Stops execution of the specified object                                                               |
| Find_Next_Object_Instance | Finds the identifier of the next unused instance of the specified object class                        |
| NOP                       | Triggers corresponding NOP response from the specified object                                         |
| Apply_Attributes          | Causes pending attribute values to become active in the specified object                              |
| Save                      | Saves attributes contents of the specified object                                                     |
| Restore                   | Restores attributes contents of the specified object                                                  |
| Get_Member                | Reads member(s) information from within an attribute                                                  |
| Set_Member                | Writes member(s) information in an attribute                                                          |
| Insert_Member             | Inserts member(s) into an attribute                                                                   |

|                                  |                                                                                                                     |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------|
| Remove_Member                    | Removes member(s) from an attribute                                                                                 |
| Group_Sync                       | Verifies that each member of a group is synchronized to System Time                                                 |
| Multiple_Service_Packet          | Performs a set of services as an autonomous sequence                                                                |
| Get_Connection_Point_Member_List | Returns EPATHs and associated sizes of the value for member attributes defining the structure of a Connection Point |

Refer to IEC 61158-5-2 for detailed definition of these services.

Primitives of Common Services exchanged between UCMM and FAL User are shown in Table 275 and Table 276.

All services have the following common parameters, as shown in Table 274.

**Table 274 – Primitives issued by FAL user to FSPM**

| Common parameters        | Req | Ind  | Rsp | Cnf  |
|--------------------------|-----|------|-----|------|
| Argument                 | M   |      |     |      |
| AREP                     | M   |      |     |      |
| Local                    | S   |      |     |      |
| UCMM Record identifier   | S   |      |     |      |
| Transport identifier     | S   |      |     |      |
| Receiver/Server Local ID |     | M    |     |      |
| Path                     | M   | C    |     |      |
| Routing Path             | M   |      |     |      |
| Add. Path                | U   | U(=) |     |      |
| Port ID                  |     | M    |     |      |
| Result (+)               |     |      | S   | S(=) |
| AREP                     |     |      |     | M=   |
| Receiver/Server Local ID |     |      | M=  |      |
| Service status           |     |      | M   | M(=) |
| Result (-)               |     |      | S   | S(=) |
| AREP                     |     |      |     | M=   |
| Receiver/Server Local ID |     |      | M=  |      |
| Service status           |     |      | M   | M(=) |

Only those argument parameters additional to the common ones are shown in Table 275 and Table 276.

Additional parameters of indication service primitives are the same as those of the corresponding request primitive.

Additional parameters of confirmation service primitives are the same as those of the corresponding response primitive.

**Table 275 – Primitives issued by FAL user to FSPM**

| Primitive name                | Source   | Additional parameters                                 | Functions                                                                                                                                              |
|-------------------------------|----------|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Get_attributes_All.req        | FAL User | None                                                  | Conveys a request from FAL User to a target object to supply values of all attributes of the specified object class or instance                        |
| Set_attributes_All.req        | FAL User | Attribute Data                                        | Conveys a request from FAL User to a target object to write specified values of all attributes of the specified object class or instance               |
| Get_attribute_list.req        | FAL User | Attribute Count<br>Attribute List                     | Conveys a request from FAL User to a target object to supply values of specified list of attributes of the specified object class or instance          |
| Set_attribute_list.req        | FAL User | Attribute Count<br>Attribute Data                     | Conveys a request from FAL User to a target object to write specified values of specified list of attributes of the specified object class or instance |
| Reset.req                     | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to reset specified target object class or instance                                                                     |
| Start.req                     | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to a target object to start an instance of the specified object class                                                  |
| Stop.req                      | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to a target object to stop an instance of the specified object class                                                   |
| Create.req                    | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to a target object to create an instance of the specified object class                                                 |
| Delete.req                    | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to a target object to delete an instance of the specified object class                                                 |
| Multiple_Service_Packet.req   | FAL User | Number of Services<br>Service Offsets<br>Service List | Conveys a request from FAL User to a target object to perform a set of services as an autonomous sequence                                              |
| Apply_Attributes.req          | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to a target object to activate pending attribute values                                                                |
| Get_attribute_single.req      | FAL User | None                                                  | Conveys a request from FAL User to a target object to supply values of the specified attribute of the specified object class or instance               |
| Set_attribute_single.req      | FAL User | Attribute Data                                        | Conveys a request from FAL User to a target object to write the specified values into specified attribute of the specified object class or instance    |
| Find_next_object_instance.req | FAL User | Maximum Returned Values                               | Conveys a request from FAL User to a target device to find the identifier of the next unused instance of the specified object class                    |
| Restore.req                   | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to a target object to restore its attributes contents                                                                  |
| Save.req                      | FAL User | Object Specific Data (optional)                       | Conveys a request from FAL User to a target object to save its attributes contents                                                                     |
| NOP.req                       | FAL User | None                                                  | Conveys a request from FAL User to a target object to send back a corresponding NOP response                                                           |
| Get_Member.req                | FAL User | None                                                  | Conveys a request from FAL User to a target object to supply member(s) information from within the specified attribute                                 |
| Set_Member.req                | FAL User | Member Data                                           | Conveys a request from FAL User to a target object to write member(s) information in the specified attribute                                           |
| Insert_Member.req             | FAL User | Member Data (optional)                                | Conveys a request from FAL User to a target object to inserts member(s) into the specified attribute                                                   |
| Remove_Member.req             | FAL User | None                                                  | Conveys a request from FAL User to a target object to removes member(s) from the specified attribute                                                   |

| Primitive name                       | Source   | Additional parameters                                                                    | Functions                                                                                                                                                                |
|--------------------------------------|----------|------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Group_Sync.req                       | FAL User | Object Specific Data (optional)                                                          | Conveys a request from FAL User to a target object to verify that each member of a group is synchronized to System Time                                                  |
| Get_Connection_Point_Member_List.req | FAL User | None                                                                                     | Conveys a request from FAL User to a target object to return EPATHs and associated sizes of the value for member attributes defining the structure of a Connection Point |
| Get_attributes_All.rsp               | FAL User | (+) Attribute Data<br>(-) Specific Status Code                                           | Returns a response from FAL User to a target object to supply values of all attributes of the specified object class or instance                                         |
| Set_attributes_All.rsp               | FAL User | (+) None<br>(-) Specific Status Code                                                     | Returns a response from FAL User to a target object to write specified values of all attributes of the specified object class or instance                                |
| Get_attribute_list.rsp               | FAL User | (+) Attribute Count<br>Attribute Data<br>(-) Specific Status Code                        | Returns a response from FAL User to a target object to supply values of specified list of attributes of the specified object class or instance                           |
| Set_attribute_list.rsp               | FAL User | (+) Attribute_count<br>Attribute Status List<br>(-) Specific Status Code                 | Returns a response from FAL User to a target object to write specified values of specified list of attributes of the specified object class or instance                  |
| Reset.rsp                            | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to confirm that the instance of the specified object class has been reset                                            |
| Start.rsp                            | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to confirm that the instance of the specified object class has started                                               |
| Stop.rsp                             | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to confirm that the instance of the specified object class has stopped                                               |
| Create.rsp                           | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to confirm that the instance of the specified object class has been created                                          |
| Delete.rsp                           | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to confirm that the instance of the specified object class has been deleted                                          |
| Multiple_Service_Packet.rsp          | FAL User | (+) Number of Responses<br>Response Offsets<br>Response List<br>(-) Specific Status Code | Returns a response from FAL User to a target object containing responses to the set of services performed as an autonomous sequence                                      |
| Apply_Attributes.rsp                 | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to confirm that pending attribute values have been activated                                                         |
| Get_attribute_single.rsp             | FAL User | (+) Attribute Data<br>(-) Specific Status Code                                           | Returns a response from FAL User to a target object to supply values of the specified attribute of the specified object class or instance                                |
| Set_attribute_single.rsp             | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to write the specified values into specified attribute of the specified object class or instance                     |
| Find_next_object_instance.rsp        | FAL User | (+) Number Of List Members<br>Instance ID List<br>(-) Specific Status Code               | Returns a response from FAL User to a target device to find the identifier of the next unused instance of the specified object class                                     |

| Primitive name                       | Source   | Additional parameters                                                             | Functions                                                                                                                                                                  |
|--------------------------------------|----------|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Restore.rsp                          | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                   | Returns a response from FAL User to a target object to confirm that its attributes contents have been restored                                                             |
| Save.rsp                             | FAL User | (+) Object Specific Data (optional)<br>(-) Specific Status Code                   | Returns a response from FAL User to a target object to confirm that its attributes contents have been saved                                                                |
| NOP.rsp                              | FAL User | None                                                                              | Returns a response from FAL User to a target object to confirm that the NOP has been received                                                                              |
| Get_Member.rsp                       | FAL User | (+) Member ID<br>Member Data<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to supply member(s) information from within the specified attribute                                                    |
| Set_Member.rsp                       | FAL User | (+) Member ID (conditional)<br>Member Data (optional)<br>(-) Specific Status Code | Returns a response from FAL User to a target object to write member(s) information in the specified attribute                                                              |
| Insert_Member.rsp                    | FAL User | (+) Member ID<br>Member Data (optional)<br>(-) Specific Status Code               | Returns a response from FAL User to a target object to inserts member(s) into the specified attribute                                                                      |
| Remove_Member.rsp                    | FAL User | (+) Member ID<br>Member Data<br>(-) Specific Status Code                          | Returns a response from FAL User to a target object to removes member(s) from the specified attribute                                                                      |
| Group_Sync.rsp                       | FAL User | (+) IsSynchronized<br>Object Specific Data (optional)<br>(-) Specific Status Code | Returns a response from FAL User to a target object to verify that each member of a group is synchronized to System Time                                                   |
| Get_Connection_Point_Member_List_rsp | FAL User | (+) Item Count<br>Members<br>(-) Specific Status Code                             | Returns a response from FAL User to a target object to provide EPATHs and associated sizes of the value for member attributes defining the structure of a Connection Point |

Table 276 – Primitives issued by FSPM to FAL user

| Primitive name              | Source | Additional parameters     | Functions                                          |
|-----------------------------|--------|---------------------------|----------------------------------------------------|
| Get_attributes_All.ind      | FSPM   | same as in .req primitive | Indicates reception of Get_attributes_All.req      |
| Set_attributes_All.ind      | FSPM   | same as in .req primitive | Indicates reception of Set_attributes_All.req      |
| Get_attribute_list.ind      | FSPM   | same as in .req primitive | Indicates reception of Get_attribute_list.req      |
| Set_attribute_list.ind      | FSPM   | same as in .req primitive | Indicates reception of Set_attribute_list.req      |
| Reset.ind                   | FSPM   | same as in .req primitive | Indicates reception of Reset.req                   |
| Start.ind                   | FSPM   | same as in .req primitive | Indicates reception of Start.req                   |
| Stop.ind                    | FSPM   | same as in .req primitive | Indicates reception of Stop.req                    |
| Create.ind                  | FSPM   | same as in .req primitive | Indicates reception of Create.req                  |
| Delete.ind                  | FSPM   | same as in .req primitive | Indicates reception of Delete.req                  |
| Multiple_Service_Packet.ind | FSPM   | same as in .req primitive | Indicates reception of Multiple_Service_Packet.req |
| Apply_Attributes.ind        | FSPM   | same as in .req primitive | Indicates reception of Apply_Attributes.req        |
| Get_attribute_single.ind    | FSPM   | same as in .req primitive | Indicates reception of Get_attribute_single.req    |
| Set_attribute_single.ind    | FSPM   | same as in .req primitive | Indicates reception of Set_attribute_single.req    |

| Primitive name                       | Source | Additional parameters     | Functions                                                   |
|--------------------------------------|--------|---------------------------|-------------------------------------------------------------|
| Find_next_object_instance.ind        | FSPM   | same as in .req primitive | Indicates reception of Find_next_object_instance.req        |
| Restore.ind                          | FSPM   | same as in .req primitive | Indicates reception of Restore.req                          |
| Save.ind                             | FSPM   | same as in .req primitive | Indicates reception of Save.req                             |
| NOP.ind                              | FSPM   | same as in .req primitive | Indicates reception of NOP.req                              |
| Get_Member.ind                       | FSPM   | same as in .req primitive | Indicates reception of Get_Member.req                       |
| Set_Member.ind                       | FSPM   | same as in .req primitive | Indicates reception of Set_Member.req                       |
| Insert_Member.ind                    | FSPM   | same as in .req primitive | Indicates reception of Insert_Member.req                    |
| Remove_Member.ind                    | FSPM   | same as in .req primitive | Indicates reception of Remove_Member.req                    |
| Group_Sync.ind                       | FSPM   | same as in .req primitive | Indicates reception of Group_Sync.req                       |
| Get_Connection_Point_Member_List.ind | FSPM   | same as in .req primitive | Indicates reception of Get_Connection_Point_Member_List.req |
| Get_attributes_All.cnf               | FSPM   | same as in .rsp primitive | Indicates reception of Get_attributes_All.rsp               |
| Set_attributes_All.cnf               | FSPM   | same as in .rsp primitive | Indicates reception of Set_attributes_All.rsp               |
| Get_attribute_list.cnf               | FSPM   | same as in .rsp primitive | Indicates reception of Get_attribute_list.rsp               |
| Set_attribute_list.cnf               | FSPM   | same as in .rsp primitive | Indicates reception of Set_attribute_list.rsp               |
| Reset.cnf                            | FSPM   | same as in .rsp primitive | Indicates reception of Reset.rsp                            |
| Start.cnf                            | FSPM   | same as in .rsp primitive | Indicates reception of Start.rsp                            |
| Stop.cnf                             | FSPM   | same as in .rsp primitive | Indicates reception of Stop.rsp                             |
| Create.cnf                           | FSPM   | same as in .rsp primitive | Indicates reception of Create.rsp                           |
| Delete.cnf                           | FSPM   | same as in .rsp primitive | Indicates reception of Delete.rsp                           |
| Multiple_Service_Packet.cnf          | FSPM   | same as in .rsp primitive | Indicates reception of Multiple_Service_Packet.rsp          |
| Apply_Attributes.cnf                 | FSPM   | same as in .rsp primitive | Indicates reception of Apply_Attributes.rsp                 |
| Get_attribute_single.cnf             | FSPM   | same as in .rsp primitive | Indicates reception of Get_attribute_single.rsp             |
| Set_attribute_single.cnf             | FSPM   | same as in .rsp primitive | Indicates reception of Set_attribute_single.rsp             |
| Find_next_object_instance.cnf        | FSPM   | same as in .rsp primitive | Indicates reception of Find_next_object_instance.rsp        |
| Restore.cnf                          | FSPM   | same as in .rsp primitive | Indicates reception of Restore.rsp                          |
| Save.cnf                             | FSPM   | same as in .rsp primitive | Indicates reception of Save.rsp                             |
| NOP.cnf                              | FSPM   | same as in .rsp primitive | Indicates reception of NOP.rsp                              |
| Get_Member.cnf                       | FSPM   | same as in .rsp primitive | Indicates reception of Get_Member.rsp                       |
| Set_Member.cnf                       | FSPM   | same as in .rsp primitive | Indicates reception of Set_Member.rsp                       |
| Insert_Member.cnf                    | FSPM   | same as in .rsp primitive | Indicates reception of Insert_Member.rsp                    |
| Remove_Member.cnf                    | FSPM   | same as in .rsp primitive | Indicates reception of Remove_Member.rsp                    |
| Group_Sync.cnf                       | FSPM   | same as in .rsp primitive | Indicates reception of Group_Sync.rsp                       |
| Get_Connection_Point_Member_List.cnf | FSPM   | same as in .rsp primitive | Indicates reception of Get_Connection_Point_Member_List.rsp |

### 8.3 Parameters of primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in Table 277.

**Table 277 – Parameters used with primitives exchanged between FAL user and FSPM**

| Parameter name                                                                 | Description                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AREP:<br>Local identifier<br>UCMM Record identifier<br>Transport identifier    | Identifies the entity to be used to convey the service. This parameter may use a dedicated identifier associated with a local entity, the identifier of a UCMM transaction record previously created, or the transport identifier returned by the connection establishment process and associated with the selected AR. |
| Receiver/Server Local ID                                                       | Generated by the Message Router ASE of the responding node. Identifies locally this invocation of the service. It is used to associate service responses with indications.                                                                                                                                              |
| Path:<br>Routing Path<br>Additional path                                       | In the request, it specifies the FAL APO or FAL APO element that is the destination of the service request.<br>In the indication, it contains only those segments beyond the logical class segment from the service request, i.e. the optional additional information for the target APO (Add.Path).                    |
| Port ID                                                                        | Indicates on which port of the device the service indication arrived.                                                                                                                                                                                                                                                   |
| Service status:<br>Status Code (mandatory)<br>Extended Status<br>(conditional) | Provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -).<br>Status code indicates the type of error (see IEC 61158-5-2 for details).<br>Extended status code gives details of the status (see IEC 61158-5-2 for details).                        |
| Attribute Data                                                                 | 1) A stream of information containing the values of each attribute that the Class/Object defines for the request/response format. Classes/Objects which support this service shall define the format of this parameter.<br>2) An array of structures, predefined by the system for the given service                    |
| Attribute Count                                                                | Number of attribute numbers in the attribute list                                                                                                                                                                                                                                                                       |
| Attribute list []                                                              | List (array) of the attribute numbers of the attributes to get from the class or object                                                                                                                                                                                                                                 |
| Attribute Number                                                               | Number representing the attribute value                                                                                                                                                                                                                                                                                 |
| Attribute Status                                                               | Status information of attribute                                                                                                                                                                                                                                                                                         |
| Attribute Value                                                                | Sequence of data specific to the attribute of the object or class                                                                                                                                                                                                                                                       |
| Object Specific Data                                                           | Class/Instance specific parameters. Class/Instance specification shall specify the format.                                                                                                                                                                                                                              |
| Instance ID                                                                    | Value assigned to identify the newly created object                                                                                                                                                                                                                                                                     |
| Maximum Returned Values                                                        | Maximum number of Instance ID values to be returned in the response message                                                                                                                                                                                                                                             |
| Number Of List Members                                                         | Number of Instance IDs specified in response message                                                                                                                                                                                                                                                                    |
| Instance ID List                                                               | Returned Instance ID List. The Instance IDs are returned in 16 bit integer fields.                                                                                                                                                                                                                                      |
| IsSynchronized                                                                 | Indicates if object is synchronized to the PTP Time Master                                                                                                                                                                                                                                                              |
| Item Count                                                                     | Number of STRUCTs returned                                                                                                                                                                                                                                                                                              |
| Members                                                                        | List of EPATHs of each member of the Connection Point structure, or the EPATH to the full Parameter Object Instance that refers to the member                                                                                                                                                                           |

## 8.4 FSPM state machines

FSPM State Machine has got only one possible state: Running.

## 9 Application relationship protocol machines (ARPMs)

### 9.1 General

This fieldbus has Application Relationship Protocol Machines for:

- connection-less application relationships,
- connection-oriented application relationships.

Connection-less relationship is of one type only.

Connection-oriented relationships fit into one of 4 transport classes:

- 0 Null (or Base)
- 1 Duplicate Detection
- 2 Acknowledged
- 3 Verified

Although similar, each of these transport classes has its own ARPM.

## 9.2 Connection-less ARPM (UCMM)

### 9.2.1 General

Functions of the connection-less ARPM are provided by the Unconnected Message Manager (UCMM) object. UCMM shall provide an unconnected request/response message services, limited to a single link that supports multiple outstanding messages. The required number of outstanding messages shall be implementation specific. The UCMM shall be present in all nodes, and shall support at least one outstanding message.

The UCMM detailed specification varies depending on the different data link layer associated which can be associated with the Type 2 application layer. The underlying data link layer defines how the UCMM is accessed and may limit the messaging which can occur across the UCMM.

### 9.2.2 Primitive definitions

The UCMM object shall provide the following services:

|             |                                                                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UCMM_Create | Creates an instance of transaction record. Puts UCMM into Running state. This service is local, it does not result in a PDU being sent.                                                           |
| UCMM_Delete | Deletes existing transaction record. Puts UCMM into Inactive state. This service is local, it does not result in a PDU being sent. This service is local, it does not result in a PDU being sent. |
| UCMM_Write  | Writes an item of application data into Transport PDU buffer; this results in sending the data to a specified target.                                                                             |
| UCMM_Abort  | Aborts existing transaction.                                                                                                                                                                      |

Refer to IEC 61158-5-2 for detailed definition of these services.

Primitives exchanged between UCMM and FSPM are shown in the following Table 278 and Table 279.

**Table 278 – Primitives issued by FSPM to ARPM**

| Primitive name  | Source | Associated parameters                                                                                    | Functions                                                                                                                                           |
|-----------------|--------|----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| UCMM_Create_req | FSPM   | fixed tag<br>max retries,                                                                                | Conveys a request from the FSPM to the ARPM to create a transaction record.                                                                         |
| UCMM_Delete_req | FSPM   | record                                                                                                   | Conveys a request from the FSPM to the ARPM to delete previously established transaction record.                                                    |
| UCMM_Write_req  | FSPM   | record<br>destination ID<br>UCMM service<br>response timeout<br>transaction priority<br>application data | Conveys a request from the FSPM to the ARPM to send an item of application data using a previously created transaction record.                      |
| UCMM_Write_rsp  | FSPM   | record<br>application data                                                                               | Conveys a response from the FSPM, via Message Router to the ARPM to send a response to UCMM_Send_req using a previously created transaction record. |
| UCMM_Abort_req  | FSPM   | record                                                                                                   | Aborts the transaction corresponding to the specified transaction record; removes the packet from the local DLL if it has not yet been transmitted. |

**Table 279 – Primitives issued by ARPM to FSPM**

| Primitive name  | Source | Associated parameters                        | Functions                                                                                                                                                                                                          |
|-----------------|--------|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UCMM_Create_cnf | ARPM   | record<br>service_status                     | Conveys a confirmation from the ARPM to the FSPM that transaction record has been created.                                                                                                                         |
| UCMM_Write_ind  | ARPM   | record<br>source ID<br>application data      | Conveys an indication from the ARPM to the Message Router that data has arrived on the previously created transaction record. The Message Router then passes the indication to the appropriate application object. |
| UCMM_Write_cnf  | ARPM   | record<br>service status<br>application data | Conveys a confirmation from the ARPM to the Message Router of the execution of FSPM UCMM_Send_req on a previously created transaction record.                                                                      |

### 9.2.3 Parameters of primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in Table 280.

**Table 280 – Parameters used with primitives exchanged between FSPM and ARPM**

| Parameter name   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| record           | Identifies unambiguously the instance of the temporary connection along which the FSPM has issued a UCMM_Send request primitive. A means for such identification is not specified in this document.                                                                                                                                                                                                                                                      |
| application data | Conveys FAL-User data.                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| fixed tag        | Set to value of 0x83 for UCMM and 0x88 for Management UCMM                                                                                                                                                                                                                                                                                                                                                                                               |
| destination ID   | Unambiguous identifier (MAC ID) of the source node of UCMM_write_req.                                                                                                                                                                                                                                                                                                                                                                                    |
| source ID        | Unambiguous identifier (MAC ID) of the destination node of UCMM_write_req.                                                                                                                                                                                                                                                                                                                                                                               |
| service          | <p>The service parameter shall specify the delivery mechanism to use for this message and shall be one of:</p> <p style="margin-left: 40px;">Request_With_Acknowledge = 2 (retries until acknowledged)<br/> Request_With_No_Response = 4 (no acknowledgement)<br/> Request_With_No_ACK = 7 (retries until response)<br/> Request_With_No_Retry = 8 (with response)</p> <p>NOTE These values correspond to command codes contained in the UCMM header</p> |
| timeout          | Duration of the transaction in microseconds. If no UCMM_Send_cnf is received before the time-out expires, the transaction is aborted and the send_status parameter shall be TIMEOUT.                                                                                                                                                                                                                                                                     |
| retries          | Conveys the maximum allowable number of retries                                                                                                                                                                                                                                                                                                                                                                                                          |
| create_status    | <p>Status returned with UCMM_Create_cnf =</p> <ul style="list-style-type: none"> <li>– success</li> <li>– cannot create</li> </ul>                                                                                                                                                                                                                                                                                                                       |
| send_status      | <p>Status returned with UCMM_Send_cnf =</p> <ul style="list-style-type: none"> <li>– success</li> <li>– record_not_created</li> <li>– packet_too_big</li> <li>– no_acknowledge</li> <li>– aborted</li> <li>– timeout</li> </ul>                                                                                                                                                                                                                          |

## 9.2.4 UCMM state machines

### 9.2.4.1 UCMM client

#### 9.2.4.1.1 States

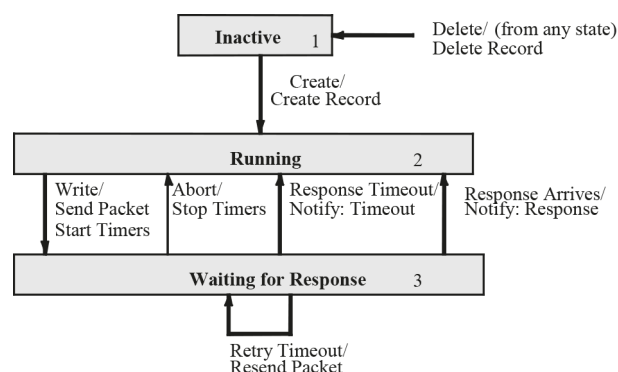
The defined states and their descriptions of the UCMM Client are listed in Table 281.

**Table 281 – UCMM client states**

| State                       | Description                                                            |
|-----------------------------|------------------------------------------------------------------------|
| <b>Inactive</b>             | The record for UCMM transfer does not exist.                           |
| <b>Running</b>              | The record for UCMM transfer has been created.                         |
| <b>Waiting for response</b> | Client sent an item of application data and is waiting for a response. |

#### 9.2.4.1.2 State transition diagram

Figure 32 shows the state transition diagram of UCMM client.



IEC

Figure 32 – State transition diagram of UCMM client

## 9.2.4.1.3 State event matrix

The state transitions for the UCMM Client shall be as specified in Table 282.

Table 282 – State event matrix of UCMM client

| Event                                                    | State                                        |                                                                                                                            |                                                                                                                                                                                                                                                   |
|----------------------------------------------------------|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                          | Inactive                                     | Running                                                                                                                    | Waiting for response                                                                                                                                                                                                                              |
| Create.req                                               | 1) create record<br>2) transition to Running |                                                                                                                            |                                                                                                                                                                                                                                                   |
| Delete.req                                               | no action                                    | 1) delete record<br>2) increment SEQUENCE<br>3) transition to Inactive                                                     | 1) Delete record<br>2) increment SEQUENCE<br>3) transition to Inactive                                                                                                                                                                            |
| Write.req                                                | confirm<br>status = INVALID_RECORD           | 1) send packet<br>2) start Response Timer<br>3) initialize Retry Count<br>4) start Retry Timer<br>5) transition to Waiting | confirm status = BUSY                                                                                                                                                                                                                             |
| Abort.req                                                | confirm<br>status = INVALID_RECORD           | no action                                                                                                                  | 1) stop timers<br>2) increment SEQUENCE<br>3) transition to Running                                                                                                                                                                               |
| Retry Timeout<br>Request packet<br>available             |                                              |                                                                                                                            | 1) Decrement Retry Count<br>IF Retry Count expired<br>2) Notify: Timeout-No ACK<br>3) Free request buffer<br>4) Increment Sequence #<br>5) Stop Response Timer<br>6) Transition to Running<br>ELSE.<br>1) Resend packet<br>2) Restart Retry Timer |
| Response<br>Timeout                                      |                                              |                                                                                                                            | 1) confirm with status = TIMEOUT-No Response<br>2) increment SEQUENCE<br>3) transition to Running                                                                                                                                                 |
| Response<br>Arrives                                      |                                              | no action                                                                                                                  | 1) confirm with status = SUCCESS<br>2) Send ACK_RESP, unless<br>response indicates no<br>ACK_RESP desired<br>3) increment SEQUENCE<br>4) transition to Running                                                                                    |
| ACK_REQ Arrives, sequence<br>number matches stored value |                                              | No Action                                                                                                                  | 1) Free request buffer<br>2) stop retry timer                                                                                                                                                                                                     |

| Event                                                      | State    |           |                      |
|------------------------------------------------------------|----------|-----------|----------------------|
|                                                            | Inactive | Running   | Waiting for response |
| ACK_REQ Arrives, sequence number not equal to stored value |          | No Action | No Action            |

The sequence number shall be set to zero at initialization. The sequence number value shall be maintained in the inactive state, to be used when the record transitions to running. The retry timeout value shall be fixed for the link at a value which guarantees that both the client and server nodes have an opportunity to transmit an unscheduled packet.

The response time-out is the response time provided when the record is created.

#### 9.2.4.2 High-end UCMM server

##### 9.2.4.2.1 Functions

When a packet arrives at the server an instance of a transaction shall be created if resources are available. If resources are not available the packet shall be dropped. When the instance is created, a transaction record shall be created for that instance. This record shall be active for the life of this transaction. The transaction shall end:

- when an ACK\_RESP is received after a response was sent;
- when the response time timer expires; or
- when a new packet is received after a response was sent.

##### 9.2.4.2.2 States

The defined states and their descriptions of the UCMM High-end Server are listed in Table 283.

**Table 283 – High-end UCMM server states**

| State                | Description                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Inactive             | The record for UCMM transfer does not exist.                                                                                                 |
| Waiting for Response | Data packet arrived with new Transaction ID. New record is created. The Server is waiting for response to be sent by the Server application. |
| Response sent        | Server application sent a response.                                                                                                          |

##### 9.2.4.2.3 State transitions

Figure 33 shows the high-end UCMM Server state transition diagram.

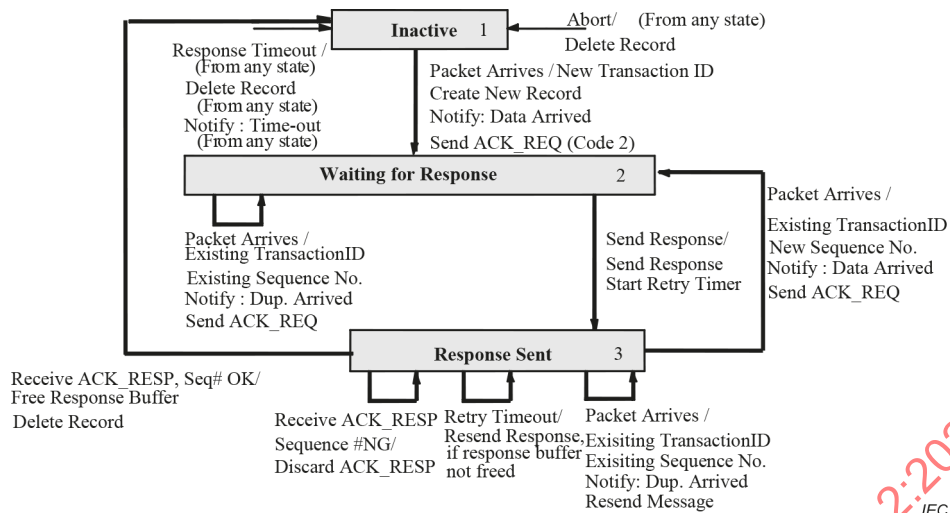


**Figure 33 – State transition diagram of high-end UCMM server**

#### 9.2.4.2.4 State event matrix

The state transitions for the high-end UCMM server shall be as specified in Table 284.

The sequence number shall be set to zero at initialization. The sequence number value shall be maintained in the Inactive state, to be used when the record transitions to Running. The retry time-out value shall be fixed for the local link.

The response time-out shall be the response time provided when the record is created.

**Table 284 – State event matrix of high-end UCMM server**

| Event                                                                       | State                                                                                                                                      |                      |                                                                                                             |
|-----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------|
|                                                                             | Inactive                                                                                                                                   | Waiting for response | Response sent                                                                                               |
| Packet arrives request (code 2)<br>New Transaction ID and source address    | 1) Create new record<br>2) Notify: Data Arrived<br>3) Send ACK_REQ<br>4) Start Response Timer<br>5) Transition to Waiting for App Response | Not Applicable       | Not Applicable                                                                                              |
| Packet arrives request (code 7)<br>New Transaction ID and source address    | 1) Create new record<br>2) Notify: Data Arrived<br>3) Start Response Timer<br>4) Transition to Waiting for App Response                    | Not Applicable       | Not Applicable                                                                                              |
| Packet arrives Existing Transaction ID and source address<br>New Sequence # | Not Applicable                                                                                                                             | No Action            | 1) Update Record<br>2) Notify: Data Arrived<br>3) Send ACK_REQ<br>4) Transition to Waiting for App Response |

| Event                                                             | State          |                                                                            |                                                                                                                                                                                                    |
|-------------------------------------------------------------------|----------------|----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                   | Inactive       | Waiting for response                                                       | Response sent                                                                                                                                                                                      |
| Packet arrives<br>Existing Transaction ID<br>Existing Sequence #  | Not Applicable | 1) Notify: Dup. Arrived (Optional)<br>2) Send ACK_REQ                      | 1) Notify: Dup. Arrived (Optional)<br>2) Resend Response Message                                                                                                                                   |
| Response Timer Expires                                            | Not Applicable | 1) Notify: Timeout<br>2) Delete record<br>3) Transition to Inactive        | 1) Notify: Timeout<br>2) Delete record<br>3) Transition to Inactive                                                                                                                                |
| Abort                                                             | Error          | 1) Delete record<br>2) Transition to Inactive                              | 1) Delete record<br>2) Transition to Inactive                                                                                                                                                      |
| Send Response                                                     | Error          | 1) Send Response<br>2) Start Retry Timer<br>3) Transition to Response Sent | Error                                                                                                                                                                                              |
| Retry Timeout                                                     | Not Applicable | Not Applicable                                                             | 1) Decrement Retry Count<br>IF Retry Count expired<br>2) Free Response buffer<br>3) Stop Response Timer<br>4) Transition to Inactive<br>ELSE<br>1) Resend Response message<br>2) Start Retry Timer |
| ACK_RESP arrives,<br>sequence number<br>matches stored value      | No action      | No action                                                                  | 1) Delete record<br>2) Transition to Inactive                                                                                                                                                      |
| ACK_RESP arrives,<br>sequence number not<br>equal to stored value | No action      | No action                                                                  | No action                                                                                                                                                                                          |

The response time is the response time received in the message header.

### 9.2.4.3 Low-end UCMM server

#### 9.2.4.3.1 Functions

The low-end server cannot support the following features:

- perform response message retries,
- detect duplicate messages.

#### 9.2.4.3.2 States

The defined states and their descriptions of the UCMM Low-end Server are listed in Table 285.

**Table 285 – Low-end UCMM server states**

| State                       | Description                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Inactive</b>             | The record for UCMM transfer does not exist.                                                                                                 |
| <b>Waiting for Response</b> | Data packet arrived with new Transaction ID. New record is created. The Server is waiting for response to be sent by the Server application. |

### 9.2.4.3.3 State transitions

Figure 34 shows the low-end UCMM server state transition diagram.

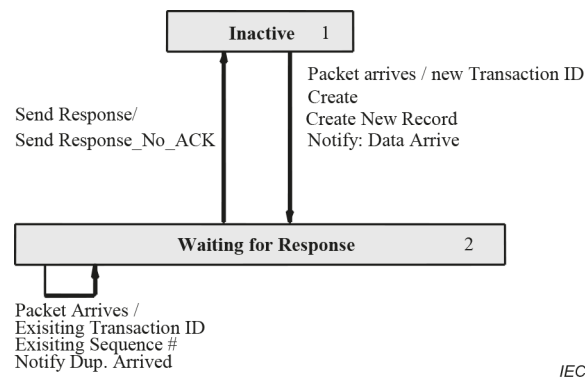


**Figure 34 – State transition diagram of low-end UCMM server**

### 9.2.4.3.4 State event matrix

The state transitions for the Low-end UCMM server shall be as specified in Table 286.

**Table 286 – State event matrix of low-end UCMM server**

| Event                                                                             | State                                                                                                                                                        |                                                      |
|-----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
|                                                                                   | Inactive                                                                                                                                                     | Waiting for response                                 |
| Packet arrives<br>New Transaction ID<br>and source address                        | 1) Create new record<br>2) Notify: Data Arrived<br>3) IF Response not immediately available,<br>Send ACK_REQ<br>4) Transition to<br>Waiting for App Response | Not Applicable                                       |
| Packet arrives<br>Existing Transaction ID<br>and source address<br>New Sequence # | No action                                                                                                                                                    | No action                                            |
| Packet arrives<br>Existing Transaction ID<br>Existing Sequence #                  | No action                                                                                                                                                    | 1) Notify: Dup. Arrived<br>(Optional)                |
| Send Response                                                                     | Error                                                                                                                                                        | 1) Send Response_No_ACK<br>2) Transition to Inactive |
| ACK_RESP arrives                                                                  | No action                                                                                                                                                    | No Action                                            |

### 9.2.5 Examples of UCMM sequences

Figure 35 shows a sequence diagram for a UCMM with one outstanding message and Figure 36 shows a sequence diagram for a UCMM with multiple outstanding messages.