

# Industrial communication networks — Fieldbus specifications —

## Part 5-10: Application layer service definition — Type 10 elements

ICS 25.040.40; 35.100.70

## National foreword

This British Standard is the UK implementation of EN 61158-5-10:2008. It is identical with IEC 61158-5-10:2007. Together with all of the other sections of BS EN 61158-5, it supersedes BS EN 61158-5:2004 which is withdrawn.

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

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

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

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

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

© BSI 2008

ISBN 978 0 580 61595 5

### Amendments/corrigenda issued since publication

| Date | Comments |
|------|----------|
|      |          |
|      |          |
|      |          |
|      |          |

**Industrial communication networks -  
Fieldbus specifications -  
Part 5-10: Application layer service definition -  
Type 10 elements  
(IEC 61158-5-10:2007)**

Réseaux de communication industriels -  
Spécifications des bus de terrain -  
Partie 5-10: Définition des services  
des couches d'application -  
Éléments de type 10  
(CEI 61158-5-10:2007)

Industrielle Kommunikationsnetze -  
Feldbusse -  
Teil 5-10: Dienstfestlegungen  
des Application Layer  
(Anwendungsschicht) -  
Typ 10-Elemente  
(IEC 61158-5-10:2007)

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

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

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

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

## CENELEC

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

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

## Foreword

The text of document 65C/475/FDIS, future edition 1 of IEC 61158-5-10, prepared by SC 65C, Industrial networks, of IEC TC 65, Industrial-process measurement, control and automation, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61158-5-10 on 2008-02-01.

This and the other parts of the EN 61158-5 series supersede EN 61158-5:2004.

With respect to EN 61158-5:2004 the following changes were made:

- deletion of Type 6 fieldbus and Type 1 fieldbus application layer, for lack of market relevance;
- addition of new fieldbus types;
- partition into multiple parts numbered 5-2, 5-3, ..., 5-20.

The following dates were fixed:

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

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

Annex ZA has been added by CENELEC.

---

## Endorsement notice

The text of the International Standard IEC 61158-5-10:2007 was approved by CENELEC as a European Standard without any modification.

---

## CONTENTS

|                                                                            |     |
|----------------------------------------------------------------------------|-----|
| INTRODUCTION .....                                                         | 17  |
| 1 Scope .....                                                              | 18  |
| 1.1 Overview .....                                                         | 18  |
| 1.2 Specifications .....                                                   | 19  |
| 1.3 Conformance .....                                                      | 19  |
| 2 Normative references .....                                               | 19  |
| 3 Terms, definitions, abbreviations, symbols and conventions .....         | 21  |
| 3.1 Referenced terms and definitions .....                                 | 21  |
| 3.2 Additional terms and definitions for distributed automation .....      | 22  |
| 3.3 Additional terms and definitions for decentralized periphery .....     | 24  |
| 3.4 Additional terms and definitions for media redundancy .....            | 32  |
| 3.5 Abbreviations and symbols .....                                        | 33  |
| 3.6 Additional abbreviations and symbols for distributed automation .....  | 34  |
| 3.7 Additional abbreviations and symbols for decentralized periphery ..... | 34  |
| 3.8 Additional abbreviations and symbols for media redundancy .....        | 35  |
| 3.9 Conventions .....                                                      | 35  |
| 4 Concepts .....                                                           | 38  |
| 5 Data type ASE .....                                                      | 38  |
| 5.1 General .....                                                          | 38  |
| 5.2 Formal definition of data type objects .....                           | 41  |
| 5.3 FAL defined data types .....                                           | 42  |
| 5.4 Data type ASE service specification .....                              | 85  |
| 6 Communication model for common services .....                            | 85  |
| 6.1 Concepts .....                                                         | 85  |
| 6.2 ASE data types .....                                                   | 85  |
| 6.3 ASEs .....                                                             | 86  |
| 7 Communication model for distributed automation .....                     | 183 |
| 7.1 Concepts .....                                                         | 183 |
| 7.2 ASE data types .....                                                   | 190 |
| 7.3 ASEs .....                                                             | 194 |
| 7.4 ARs .....                                                              | 411 |
| 7.5 Summary of FAL classes .....                                           | 415 |
| 7.6 Summary of FAL services .....                                          | 416 |
| 8 Communication model for decentralized periphery .....                    | 419 |
| 8.1 Concepts .....                                                         | 419 |
| 8.2 ASE data types .....                                                   | 435 |
| 8.3 ASEs .....                                                             | 435 |
| 8.4 Behavior of an IO device .....                                         | 670 |
| 8.5 Behavior of an IO controller .....                                     | 736 |
| 8.6 Application characteristics .....                                      | 743 |

|                                                                                                                                |     |
|--------------------------------------------------------------------------------------------------------------------------------|-----|
| Annex A (informative) Device instances .....                                                                                   | 745 |
| Annex B (informative) Components of an Ethernet interface .....                                                                | 747 |
| Annex C (informative) Scheme of MAC address assignment .....                                                                   | 751 |
| Annex D (informative) Collection of objects .....                                                                              | 752 |
| Annex E (informative) Measurement of the fast startup time .....                                                               | 753 |
| Annex ZA (normative) Normative references to international publications with their<br>corresponding European publications..... | 755 |
| Bibliography .....                                                                                                             | 754 |
| Figure 1 – Data type class hierarchy example .....                                                                             | 39  |
| Figure 2 – NetworkTime date relation .....                                                                                     | 62  |
| Figure 3 – UTF-8 coding of four different characters .....                                                                     | 66  |
| Figure 4 – PTCP applications .....                                                                                             | 99  |
| Figure 5 – Clock drift measurement .....                                                                                       | 109 |
| Figure 6 – Multiple synchronization .....                                                                                      | 109 |
| Figure 7 –MRP stack.....                                                                                                       | 111 |
| Figure 8 – Ring topology with one manager and clients.....                                                                     | 122 |
| Figure 9 – MRM in an open ring .....                                                                                           | 122 |
| Figure 10 – More than one MRM in the ring .....                                                                                | 124 |
| Figure 11 – Media redundancy diagnosis dependencies.....                                                                       | 125 |
| Figure 12 – Locating the destination for redundant RT frames .....                                                             | 164 |
| Figure 13 – Example of periods at a local port .....                                                                           | 171 |
| Figure 14 – FAL ASEs communication architecture .....                                                                          | 186 |
| Figure 15 – Runtime object model.....                                                                                          | 187 |
| Figure 16 – Relationship between engineering and runtime.....                                                                  | 188 |
| Figure 17 – Navigation in the runtime object model .....                                                                       | 189 |
| Figure 18 – Operating state block diagram.....                                                                                 | 228 |
| Figure 19 – Device status model for the common diagnosis .....                                                                 | 229 |
| Figure 20 – ACCO ASE structure .....                                                                                           | 260 |
| Figure 21 – Productive operation of data connections .....                                                                     | 261 |
| Figure 22 – Quality code transfer – standard behavior .....                                                                    | 273 |
| Figure 23 – Startup of a connection .....                                                                                      | 274 |
| Figure 24 – Quality code with communication fault.....                                                                         | 274 |
| Figure 25 – Quality code when an connection is cleared .....                                                                   | 275 |
| Figure 26 – Quality code when an connection is deactivated.....                                                                | 275 |
| Figure 27 – Quality code during the transfer of "incorrect" connection data .....                                              | 276 |
| Figure 28 – Quality code for provider in "CBARReady" state .....                                                               | 277 |
| Figure 29 – Quality code when clearing an object from the provider .....                                                       | 277 |
| Figure 30 – Quality code when a connection is forced.....                                                                      | 278 |
| Figure 31 – Quality code at QoS violation .....                                                                                | 278 |
| Figure 32 – Push mode .....                                                                                                    | 285 |

|                                                                                          |     |
|------------------------------------------------------------------------------------------|-----|
| Figure 33 – Pull mode overview .....                                                     | 286 |
| Figure 34 – Detailed sequence chart of the pull mode .....                               | 287 |
| Figure 35 – QoS and ORPC communication channel.....                                      | 288 |
| Figure 36 – QoS Violation within Pull Mode.....                                          | 289 |
| Figure 37 – Monitoring the providers heartbeat .....                                     | 290 |
| Figure 38 – State machine RemoteAcco .....                                               | 292 |
| Figure 39 – State machine RemoteAccoProvider.....                                        | 292 |
| Figure 40 – State machine RemoteAccoProvider <sub>ORPC</sub> .....                       | 293 |
| Figure 41 – State machine AR <sub>ORPC</sub> – Provider .....                            | 293 |
| Figure 42 – State machine GetConnectionData – Provider .....                             | 294 |
| Figure 43 – State machine ProviderConnection.....                                        | 294 |
| Figure 44 – State machine ProvConnActivation.....                                        | 295 |
| Figure 45 – State machine WorkerORPC .....                                               | 295 |
| Figure 46 – Communication stack of distributed automation devices .....                  | 296 |
| Figure 47 – Application relations between devices .....                                  | 297 |
| Figure 48 – Communication relations .....                                                | 297 |
| Figure 49 – RT communication channel .....                                               | 299 |
| Figure 50 – Interaction between provider and consumer .....                              | 301 |
| Figure 51 – State machine AR <sub>SRT</sub> – Consumer .....                             | 303 |
| Figure 52 – State machine AR <sub>SRT</sub> – Provider.....                              | 303 |
| Figure 53 – State machine AccoDataCR – Consumer.....                                     | 304 |
| Figure 54 – State machine AccoDataCR – Provider .....                                    | 305 |
| Figure 55 – RT frame layout.....                                                         | 306 |
| Figure 56 – Establishing an AccoDataCR.....                                              | 307 |
| Figure 57 – Flowchart of the copy cycle for local connections .....                      | 308 |
| Figure 58 – State machine connect attempt .....                                          | 312 |
| Figure 59 – Productive operation of data connections (ORPC channel).....                 | 319 |
| Figure 60 – Productive operation of data connections (RT channel) .....                  | 320 |
| Figure 61 – Productive operation of data connections (Local channel) .....               | 320 |
| Figure 62 – Data flow for cyclic RT .....                                                | 321 |
| Figure 63 – Failure of the provider in productive operation (ORPC push mode).....        | 323 |
| Figure 64 – Failure of the provider in productive operation (ORPC pull mode) .....       | 324 |
| Figure 65 – Scenario 1: Provider failure in productive operation (RT).....               | 325 |
| Figure 66 – Scenario 2: Recovery from provider failure in productive operation (RT)..... | 326 |
| Figure 67 – Failure of the consumer (push mode) .....                                    | 326 |
| Figure 68 – Failure of the consumer (pull mode) .....                                    | 327 |
| Figure 69 – Failure of the consumer.....                                                 | 328 |
| Figure 70 – Failure of the provider when setting up connections .....                    | 330 |
| Figure 71 – Information levels.....                                                      | 331 |
| Figure 72 – ACCO ASE status model for the common diagnosis .....                         | 331 |
| Figure 73 – ACCO ASE status model for the detailed diagnosis .....                       | 332 |
| Figure 74 – Structure of the transmitted connection data .....                           | 374 |

|                                                                                                                        |     |
|------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 75 – Example of communication between controlling devices and field devices .....                               | 420 |
| Figure 76 – Example of communication between an engineering station and several<br>controlling and field devices ..... | 420 |
| Figure 77 – Example of communication between field devices and a server station.....                                   | 421 |
| Figure 78 – Example of communication between field devices .....                                                       | 421 |
| Figure 79 – Structural units of one arbitrary API of an IO device (general) .....                                      | 423 |
| Figure 80 – Example 1 structural units for interfaces and ports within API 0 .....                                     | 424 |
| Figure 81 – Example 2 structural units for interfaces and ports within API 0 .....                                     | 425 |
| Figure 82 – Overview of application processes .....                                                                    | 427 |
| Figure 83 – IO device with APs, slots and subslots .....                                                               | 428 |
| Figure 84 – Application Process with application objects (APOs) .....                                                  | 431 |
| Figure 85 – Access to a remote APO .....                                                                               | 432 |
| Figure 86 – Access to a remote APO for provider/consumer association .....                                             | 433 |
| Figure 87 – Example of one AR with two AREPs .....                                                                     | 434 |
| Figure 88 – Relation of a record data object to one real object.....                                                   | 436 |
| Figure 89 – Relation of a record data object to two real objects .....                                                 | 437 |
| Figure 90 – Overview IO ASE service interactions .....                                                                 | 447 |
| Figure 91 – Example of a resource model at the alarm source .....                                                      | 524 |
| Figure 92 – General isochronous application model (example).....                                                       | 559 |
| Figure 93 – ASE relations in an IO device operating in isochronous mode .....                                          | 565 |
| Figure 94 – State machine relations in an IO device operating in isochronous mode .....                                | 565 |
| Figure 95 – SyncCtl state diagram .....                                                                                | 569 |
| Figure 96 – Output state diagram.....                                                                                  | 571 |
| Figure 97 – Input state diagram .....                                                                                  | 576 |
| Figure 98 – Assignment of communication relationship to application relationship .....                                 | 649 |
| Figure 99 – Implicit application relationship.....                                                                     | 652 |
| Figure 100 – Example IO application relationship (one-to-one) .....                                                    | 654 |
| Figure 101 – Example IO application relationship one-to-many .....                                                     | 655 |
| Figure 102 – Overview ASE state machines for IO device .....                                                           | 671 |
| Figure 103 – State diagram application startup IO device.....                                                          | 672 |
| Figure 104 – State diagram startup manager IO device.....                                                              | 680 |
| Figure 105 – State diagram IRT port state IO device (standard).....                                                    | 683 |
| Figure 106 – State diagram IRT port state IO device (optimized).....                                                   | 695 |
| Figure 107 – State diagram neighborhood check.....                                                                     | 702 |
| Figure 108 – State diagram PD parameter check IO device .....                                                          | 711 |
| Figure 109 – State diagram for a submodule.....                                                                        | 721 |
| Figure 110 – State diagram IO controller during startup .....                                                          | 737 |
| Figure 111 – State diagram startup manager IO controller .....                                                         | 740 |
| Figure 112 – Example of network topology including slower wireless segments .....                                      | 744 |
| Figure 113 – Example of media redundancy including wireless segments .....                                             | 744 |
| Figure A.1 – Instance model .....                                                                                      | 745 |

|                                                                                                       |     |
|-------------------------------------------------------------------------------------------------------|-----|
| Figure B.1 – Scheme of an Ethernet interface .....                                                    | 747 |
| Figure B.2 – Scheme of an Ethernet interface with bridging ability .....                              | 748 |
| Figure B.3 – Scheme of an Ethernet interface with optical ports .....                                 | 749 |
| Figure B.4 – Scheme of an Ethernet interface with bridging ability using radio<br>communication ..... | 750 |
| Figure B.5 – Scheme of an Ethernet interface with radio communication .....                           | 750 |
| Figure C.1 – Scheme of MAC address assignment .....                                                   | 751 |
| Figure D.1 – Example for an intersection of IO device, slot, and AR .....                             | 752 |
| Figure E-1 – Measurement of the fast startup time .....                                               | 753 |
|                                                                                                       |     |
| Table 1 – PERSISTDEF .....                                                                            | 45  |
| Table 2 – VARTYPE .....                                                                               | 46  |
| Table 3 – ITEMQUALITYDEF .....                                                                        | 46  |
| Table 4 – STATEDEF .....                                                                              | 50  |
| Table 5 – GROUPEXCEPTIONDEF .....                                                                     | 50  |
| Table 6 – ACCESSRIGHTSDEF .....                                                                       | 50  |
| Table 7 – HRESULT .....                                                                               | 51  |
| Table 8 – N2 value range .....                                                                        | 55  |
| Table 9 – N2 octets .....                                                                             | 55  |
| Table 10 – N4 value range .....                                                                       | 56  |
| Table 11 – N4 octets .....                                                                            | 56  |
| Table 12 – X2 value range .....                                                                       | 57  |
| Table 13 – X2 octets .....                                                                            | 57  |
| Table 14 – X4 value range .....                                                                       | 57  |
| Table 15 – X4 octets .....                                                                            | 57  |
| Table 16 – Unipolar2.16 value range .....                                                             | 58  |
| Table 17 – Unipolar2.16 octets .....                                                                  | 58  |
| Table 18 – E2 value range .....                                                                       | 58  |
| Table 19 – E2 octets .....                                                                            | 58  |
| Table 20 – C4 value range .....                                                                       | 59  |
| Table 21 – V2 octets .....                                                                            | 59  |
| Table 22 – L2 octets .....                                                                            | 59  |
| Table 23 – UUID for decentralized peripherals .....                                                   | 60  |
| Table 24 – UUID for distributed automation .....                                                      | 61  |
| Table 25 – NetworkTime values .....                                                                   | 62  |
| Table 26 – NetworkTime octets .....                                                                   | 62  |
| Table 27 – T2 values .....                                                                            | 63  |
| Table 28 – T4 values .....                                                                            | 63  |
| Table 29 – D2 values .....                                                                            | 64  |
| Table 30 – R2 values .....                                                                            | 64  |
| Table 31 – UNICODEString values .....                                                                 | 65  |

|                                                    |     |
|----------------------------------------------------|-----|
| Table 32 – UTF-8 character encoding scheme .....   | 66  |
| Table 33 – OctetString2+Unsigned8 octets .....     | 77  |
| Table 34 – Float32+Unsigned8 octets .....          | 78  |
| Table 35 – Unsigned8+Unsigned8 octets .....        | 78  |
| Table 36 – Data Types for Value in a VARIANT ..... | 81  |
| Table 37 – Unsigned16_S octets.....                | 82  |
| Table 38 – Unsigned16_S meaning.....               | 82  |
| Table 39 – Integer16_S octets .....                | 83  |
| Table 40 – Integer16_S meaning .....               | 83  |
| Table 41 – Unsigned8_S octets .....                | 83  |
| Table 42 – Unsigned8_S meaning.....                | 83  |
| Table 43 – OctetString_S octets .....              | 84  |
| Table 44 – OctetString_S status bits .....         | 84  |
| Table 45 – F message trailer with 4 octets .....   | 84  |
| Table 46 – F message trailer with 5 octets .....   | 85  |
| Table 47 – Get.....                                | 91  |
| Table 48 – Set .....                               | 93  |
| Table 49 – Identify .....                          | 96  |
| Table 50 – Hello.....                              | 97  |
| Table 51 – Clock stratum .....                     | 101 |
| Table 52 – Start bridge .....                      | 103 |
| Table 53 – Start slave .....                       | 104 |
| Table 54 – Start master .....                      | 105 |
| Table 55 – Stop bridge.....                        | 106 |
| Table 56 –Stop slave .....                         | 106 |
| Table 57 – Stop master.....                        | 107 |
| Table 58 – State change .....                      | 108 |
| Table 59 – Start MRM .....                         | 116 |
| Table 60 – Stop MRM .....                          | 118 |
| Table 61 – Change state.....                       | 118 |
| Table 62 – Start MRC .....                         | 119 |
| Table 63 – Stop MRC.....                           | 120 |
| Table 64 – Neighborhood changed.....               | 121 |
| Table 65 – MRP network/connection parameters ..... | 126 |
| Table 66 – MRM parameters .....                    | 126 |
| Table 67 – MRC parameters .....                    | 126 |
| Table 68 – Set prov data.....                      | 127 |
| Table 69 – Set prov status .....                   | 128 |
| Table 70 – PPM activate .....                      | 129 |
| Table 71 – Close.....                              | 130 |
| Table 72 – Start .....                             | 130 |

|                                                      |     |
|------------------------------------------------------|-----|
| Table 73 – Error .....                               | 131 |
| Table 74 – Get cons data .....                       | 131 |
| Table 75 – Get cons status .....                     | 132 |
| Table 76 – Set redRole .....                         | 132 |
| Table 77 – CPM activate .....                        | 133 |
| Table 78 – APMS activate .....                       | 137 |
| Table 79 – APMR activate .....                       | 138 |
| Table 80 – APMS A data .....                         | 139 |
| Table 81 – APMR A data .....                         | 140 |
| Table 82 – APMR ack .....                            | 140 |
| Table 83 – APMS error .....                          | 141 |
| Table 84 – APMS error ERRCLS/ERRCODE .....           | 141 |
| Table 85 – APMR error .....                          | 142 |
| Table 86 – APMR error ERRCLS/ERRCODE .....           | 142 |
| Table 87 – APMS_Close .....                          | 142 |
| Table 88 – APMR_Close .....                          | 143 |
| Table 89 – Connect .....                             | 144 |
| Table 90 – Release .....                             | 145 |
| Table 91 – Read .....                                | 146 |
| Table 92 – Write .....                               | 147 |
| Table 93 – Control .....                             | 148 |
| Table 94 – System capabilities .....                 | 153 |
| Table 95 – Auto negotiation support and status ..... | 154 |
| Table 96 – MDI Power Support .....                   | 155 |
| Table 97 – Link aggregation status .....             | 155 |
| Table 98 – Remote systems data change .....          | 158 |
| Table 99 – Allowed values of ReductionRatio .....    | 161 |
| Table 100 – Frame IDs for RT_CLASS_3 .....           | 162 |
| Table 101 – Sync Frame .....                         | 162 |
| Table 102 – FrameSendOffset .....                    | 162 |
| Table 103 – Tx Port Entry .....                      | 163 |
| Table 104 – Port state change .....                  | 166 |
| Table 105 – Set port state .....                     | 166 |
| Table 106 – Flush filtering data base .....          | 166 |
| Table 107 – IFW IRT Schedule Add .....               | 167 |
| Table 108 – IFW IRT Schedule Remove .....            | 167 |
| Table 109 – IFW Schedule .....                       | 168 |
| Table 110 – MAU type change .....                    | 172 |
| Table 111 – Set MAU type .....                       | 173 |
| Table 112 – IP Multicast address .....               | 175 |
| Table 113 – Set ARP cache .....                      | 175 |

|                                                                                 |     |
|---------------------------------------------------------------------------------|-----|
| Table 114 – Enterprise number .....                                             | 178 |
| Table 115 – Vendor OUI .....                                                    | 178 |
| Table 116 – IRT schedule sdd.....                                               | 179 |
| Table 117 – IRT schedule remove.....                                            | 180 |
| Table 118 – Schedule .....                                                      | 180 |
| Table 119 – N data .....                                                        | 181 |
| Table 120 – A data.....                                                         | 182 |
| Table 121 – C data .....                                                        | 183 |
| Table 122 – Connectable data types .....                                        | 191 |
| Table 123 – Supported data types according to the Base Object Version .....     | 192 |
| Table 124 – Usage of character sets.....                                        | 194 |
| Table 125 – QueryInterface (Unknown interface) .....                            | 196 |
| Table 126 – AddRef (Unknown interface) .....                                    | 197 |
| Table 127 – Release (Unknown interface).....                                    | 198 |
| Table 128 – GetTypeInfoCount (Dispatch interface).....                          | 199 |
| Table 129 – GetTypeInfo (Dispatch interface) .....                              | 200 |
| Table 130 – GetIDsOfNames (Dispatch interface) .....                            | 201 |
| Table 131 – Invoke (Dispatch interface) .....                                   | 202 |
| Table 132 – CRC table for the PDev stamp calculation (hexadecimal values) ..... | 207 |
| Table 133 – get_Producer (Physical device interface).....                       | 208 |
| Table 134 – get_Product (Physical device interface).....                        | 209 |
| Table 135 – get_SerialNo (Physical device interface).....                       | 210 |
| Table 136 – get_ProductionDate (Physical device interface) .....                | 211 |
| Table 137 – Revision (Physical device interface) .....                          | 212 |
| Table 138 – get_LogicalDevice (Physical device interface) .....                 | 213 |
| Table 139 – Type (Physical device interface) .....                              | 214 |
| Table 140 – PROFInetRevision (Physical device interface) .....                  | 215 |
| Table 141 – get_PDevStamp (Physical device interface).....                      | 216 |
| Table 142 – get_Count (Browse interface) .....                                  | 217 |
| Table 143 – BrowseItems (Browse interface) .....                                | 218 |
| Table 144 – get_Count2 (Browse interface) .....                                 | 219 |
| Table 145 – BrowseItems2 (Browse interface) .....                               | 221 |
| Table 146 – Save (Persist interface) .....                                      | 222 |
| Table 147 – Save2 (Persist interface) .....                                     | 223 |
| Table 148 – get_Name (Logical device interface).....                            | 229 |
| Table 149 – get_Producer (Logical device interface).....                        | 230 |
| Table 150 – get_Product (Logical device interface).....                         | 231 |
| Table 151 – get_SerialNo (Logical device interface).....                        | 232 |
| Table 152 – get_ProductionDate (Logical device interface).....                  | 233 |
| Table 153 – Revision (Logical device interface) .....                           | 234 |
| Table 154 – get_ACCO (Logical device interface) .....                           | 235 |

|                                                                   |     |
|-------------------------------------------------------------------|-----|
| Table 155 – get_RTAuto (Logical device interface) .....           | 236 |
| Table 156 – PROFINetRevision (Logical device interface) .....     | 237 |
| Table 157 – ComponentInfo (Logical device interface) .....        | 238 |
| Table 158 – get_State (State interface).....                      | 239 |
| Table 159 – Activate (State interface) .....                      | 241 |
| Table 160 – Deactivate (State interface) .....                    | 242 |
| Table 161 – Reset (State interface).....                          | 243 |
| Table 162 – AdviseState (State interface) .....                   | 244 |
| Table 163 – UnadviseState (State interface) .....                 | 245 |
| Table 164 – get_Time (Time interface).....                        | 246 |
| Table 165 – put_Time (Time interface).....                        | 247 |
| Table 166 – get_Count (Browse interface) .....                    | 248 |
| Table 167 – BrowseItems (Browse interface) .....                  | 249 |
| Table 168 – get_Count2 (Browse interface) .....                   | 250 |
| Table 169 – BrowseItems2 (Browse interface) .....                 | 252 |
| Table 170 – GroupError (Group error interface) .....              | 253 |
| Table 171 – AdviseGroupError (Group Error interface).....         | 255 |
| Table 172 – UnadviseGroupError (Group Error interface).....       | 256 |
| Table 173 – PingFactor values.....                                | 258 |
| Table 174 – QoS subtypes in the ORPC communication channel .....  | 267 |
| Table 175 – QoS subtypes in the RT communication channel .....    | 268 |
| Table 176 – QoS Types and Values .....                            | 268 |
| Table 177 – Epsilon value for connectable data types .....        | 270 |
| Table 178 – Quality Codes .....                                   | 272 |
| Table 179 – Quality code priority table .....                     | 280 |
| Table 180 – Maximum ORPC substitute value apply time.....         | 290 |
| Table 181 – Maximum GetConnectionData hold time .....             | 291 |
| Table 182 – Usage of RT Variants .....                            | 298 |
| Table 183 – Mapping QoS Value to RT cycle time.....               | 299 |
| Table 184 – Maximum RT Substitute Value Apply Time .....          | 300 |
| Table 185 – Time Intervals and Timeouts.....                      | 322 |
| Table 186 – Error codes for the ACCO ASE detailed diagnosis ..... | 333 |
| Table 187 – AddConnections (ACCO management interface).....       | 335 |
| Table 188 – RemoveConnections (ACCO management interface) .....   | 336 |
| Table 189 – ClearConnections (ACCO management interface) .....    | 338 |
| Table 190 – SetActivationState (ACCO management interface) .....  | 339 |
| Table 191 – GetInfo (ACCO management interface) .....             | 340 |
| Table 192 – GetIDs (ACCO management interface) .....              | 341 |
| Table 193 – GetConnections (ACCO management interface) .....      | 342 |
| Table 194 – ReviseQoS (ACCO management interface).....            | 344 |
| Table 195 – get_PingFactor (ACCO management interface) .....      | 345 |

|                                                                                       |     |
|---------------------------------------------------------------------------------------|-----|
| Table 196 – put_PingFactor (ACCO management interface) .....                          | 346 |
| Table 197 – get_CDBCcookie (ACCO management interface) .....                          | 347 |
| Table 198 – GetConsIDs (ACCO management interface) .....                              | 348 |
| Table 199 – GetConsConnections (ACCO management interface) .....                      | 349 |
| Table 200 – DiagConsConnections (ACCO management interface).....                      | 350 |
| Table 201 – GetProvIDs (ACCO management interface) .....                              | 351 |
| Table 202 – GetProvConnections (ACCO management interface) .....                      | 353 |
| Table 203 – GetDiagnosis (ACCO management interface) .....                            | 354 |
| Table 204 – Request.....                                                              | 356 |
| Table 205 – Connect (ACCO server interface) .....                                     | 363 |
| Table 206 – Disconnect (ACCO server interface) .....                                  | 365 |
| Table 207 – DisconnectMe (ACCO server interface) .....                                | 366 |
| Table 208 – SetActivation (ACCO server interface).....                                | 367 |
| Table 209 – Ping (ACCO server interface) .....                                        | 368 |
| Table 210 – Connect2 (ACCO server interface) .....                                    | 370 |
| Table 211 – GetConnectionData (ACCO server interface).....                            | 372 |
| Table 212 – OnDataChanged (ACCO callback interface).....                              | 375 |
| Table 213 – Version .....                                                             | 375 |
| Table 214 – Flags .....                                                               | 376 |
| Table 215 – Gnip (ACCO callback interface).....                                       | 377 |
| Table 216 – ReadItems (ACCO sync interface) .....                                     | 378 |
| Table 217 – WriteItems (ACCO sync interface) .....                                    | 380 |
| Table 218 – WriteItemsQCD (ACCO sync interface).....                                  | 382 |
| Table 219 – GroupError (Group Error interface) .....                                  | 383 |
| Table 220 – AdviseGroupError (Group Error interface).....                             | 384 |
| Table 221 – UnadviseGroupError (Group Error interface).....                           | 385 |
| Table 222 – ConnectCR (ACCO server SRT interface).....                                | 387 |
| Table 223 – DisconnectCR (ACCO server SRT interface) .....                            | 389 |
| Table 224 – Connect (ACCO server SRT interface).....                                  | 390 |
| Table 225 – Disconnect (ACCO server SRT interface) .....                              | 391 |
| Table 226 – DisconnectMe (ACCO server SRT interface).....                             | 393 |
| Table 227 – SetActivation (ACCO server SRT interface) .....                           | 394 |
| Table 228 –Hresult values for access to properties of Custom RT-Auto objects .....    | 399 |
| Table 229 – Hresult values for access to properties of the System RT-Auto object..... | 400 |
| Table 230 – Common hresult values on access to properties of RT-Auto objects .....    | 400 |
| Table 231 – Quality code for access to properties of Custom RT-Auto objects .....     | 400 |
| Table 232 – Quality code for access to properties of the System RT-Auto object .....  | 401 |
| Table 233 – get_Name (RT-Auto interface) .....                                        | 401 |
| Table 234 – Revision (RT-Auto interface).....                                         | 402 |
| Table 235 – ComponentInfo (RT-Auto interface) .....                                   | 403 |
| Table 236 – get_Count (Browse interface) .....                                        | 404 |

|                                                                                                |     |
|------------------------------------------------------------------------------------------------|-----|
| Table 237 – BrowseItems (Browse interface) .....                                               | 405 |
| Table 238 – get_Count2 (Browse interface) .....                                                | 407 |
| Table 239 – BrowseItems2 (Browse interface) .....                                              | 408 |
| Table 240 – get_StateCollection (System properties interface) .....                            | 410 |
| Table 241 – get_StampCollection (System Properties interface) .....                            | 411 |
| Table 242 – CoCreateInstance.....                                                              | 413 |
| Table 243 – CoDisconnectObject .....                                                           | 414 |
| Table 244 – Call.....                                                                          | 414 |
| Table 245 – Distributed automation FAL class summary .....                                     | 415 |
| Table 246 – Assignment of the services to client and server .....                              | 416 |
| Table 247 – Requirements and features.....                                                     | 419 |
| Table 248 – Persistence behavior for record data objects .....                                 | 439 |
| Table 249 – Read .....                                                                         | 440 |
| Table 250 – Read query.....                                                                    | 442 |
| Table 251 – Write .....                                                                        | 444 |
| Table 252 – Set input.....                                                                     | 454 |
| Table 253 – Set input IOCS .....                                                               | 455 |
| Table 254 – Get input .....                                                                    | 456 |
| Table 255 – Get input IOCS .....                                                               | 457 |
| Table 256 – New input .....                                                                    | 458 |
| Table 257 – Set input APDU data status .....                                                   | 459 |
| Table 258 – New input APDU data status.....                                                    | 460 |
| Table 259 – Read input data .....                                                              | 461 |
| Table 260 – Set output.....                                                                    | 462 |
| Table 261 – Set output IOCS .....                                                              | 464 |
| Table 262 – Get output .....                                                                   | 465 |
| Table 263 – Get output IOCS.....                                                               | 466 |
| Table 264 – New output .....                                                                   | 467 |
| Table 265 – Set output APDU data status .....                                                  | 468 |
| Table 266 – New output APDU data status .....                                                  | 469 |
| Table 267 – Read output data .....                                                             | 470 |
| Table 268 – Read output substitute data .....                                                  | 473 |
| Table 269 – Write output substitute data .....                                                 | 475 |
| Table 270 – Read logbook .....                                                                 | 479 |
| Table 271 – Logbook event.....                                                                 | 480 |
| Table 272 – Dependencies within channel properties.....                                        | 484 |
| Table 273 – Ext channel error type .....                                                       | 487 |
| Table 274 – Ext channel add value for accumulative info .....                                  | 488 |
| Table 275 – Dependencies within channel properties for manufacturer specific<br>diagnosis..... | 489 |
| Table 276 – Read device diagnosis.....                                                         | 491 |

|                                                                                |     |
|--------------------------------------------------------------------------------|-----|
| Table 277 – Diagnosis item .....                                               | 493 |
| Table 278 – Diagnosis event .....                                              | 499 |
| Table 279 – State table diagnosis entry .....                                  | 502 |
| Table 280 – Functions used in state tables .....                               | 503 |
| Table 281 – State table maintenance required entry .....                       | 504 |
| Table 282 – State table maintenance demanded entry .....                       | 505 |
| Table 283 – State table qualified entry .....                                  | 506 |
| Table 284 – Alarm type .....                                                   | 511 |
| Table 285 – Channel diagnosis .....                                            | 512 |
| Table 286 – Manufacturer specific diagnosis .....                              | 512 |
| Table 287 – Submodule diagnosis state .....                                    | 512 |
| Table 288 – AR diagnosis state .....                                           | 513 |
| Table 289 – User structure Identifier .....                                    | 514 |
| Table 290 – Semantics of specifier .....                                       | 515 |
| Table 291 – Alarm notification .....                                           | 520 |
| Table 292 – Alarm ack .....                                                    | 523 |
| Table 293 – Module State .....                                                 | 528 |
| Table 294 – Usage with respect to CR type .....                                | 530 |
| Table 295 – Detail .....                                                       | 531 |
| Table 296 – ARInfo .....                                                       | 532 |
| Table 297 – Ident Info .....                                                   | 532 |
| Table 298 – Connect .....                                                      | 533 |
| Table 299 – Connect device access .....                                        | 540 |
| Table 300 – Release .....                                                      | 542 |
| Table 301 – Abort .....                                                        | 543 |
| Table 302 – End of parameter .....                                             | 543 |
| Table 303 – Application ready .....                                            | 544 |
| Table 304 – Ready for companion .....                                          | 546 |
| Table 305 – Read expected identification .....                                 | 547 |
| Table 306 – Read real identification .....                                     | 550 |
| Table 307 – Read identification difference .....                               | 553 |
| Table 308 – Write IsoM data .....                                              | 560 |
| Table 309 – Read IsoM data .....                                               | 562 |
| Table 310 – SYNCH event .....                                                  | 564 |
| Table 311 – Primitives issued by the AL to the SyncCtl state machine .....     | 566 |
| Table 312 – Primitive issued by the SyncCtl state machine to the user .....    | 567 |
| Table 313 – Primitives issued by the input state machine to the user .....     | 567 |
| Table 314 – Primitive issued by the output state machine to the user .....     | 567 |
| Table 315 – Primitives issued by the SyncCtl to the output state machine ..... | 567 |
| Table 316 – Primitives issued by the output to the SyncCtl state machine ..... | 567 |
| Table 317 – Primitives issued by the SyncCtl to the input state machine .....  | 568 |

|                                                                           |     |
|---------------------------------------------------------------------------|-----|
| Table 318 – Primitives issued by the output state machine to the AL ..... | 568 |
| Table 319 – Primitives issued by the AL to the output state machine ..... | 568 |
| Table 320 – Primitives issued by the input state machine to the AL.....   | 568 |
| Table 321 – Primitives issued by the AL to the input state machine.....   | 568 |
| Table 322 – SyncCtl state table .....                                     | 570 |
| Table 323 – Output state table .....                                      | 572 |
| Table 324 – Input state table.....                                        | 577 |
| Table 325 – Subslot number for interface submodules .....                 | 582 |
| Table 326 – Subslot number for port submodules.....                       | 582 |
| Table 327 – Subslot Number for Interface Submodules .....                 | 584 |
| Table 328 – Subslot Number for Sync Interface Submodules .....            | 584 |
| Table 329 – Sync Properties Role .....                                    | 585 |
| Table 330 – Sync Class .....                                              | 586 |
| Table 331 – Subslot Number for Fiber Optic Submodules .....               | 586 |
| Table 332 – Fiber Optic Types .....                                       | 587 |
| Table 333 – Fiber Optic Cable Types .....                                 | 587 |
| Table 334 – Write expected port data.....                                 | 590 |
| Table 335 – Write adjusted port data .....                                | 593 |
| Table 336 – Read real port data.....                                      | 595 |
| Table 337 – Read expected port data .....                                 | 598 |
| Table 338 – Read adjusted port data .....                                 | 600 |
| Table 339 – Write IR data .....                                           | 603 |
| Table 340 – Read IR data .....                                            | 607 |
| Table 341 – Write sync data.....                                          | 611 |
| Table 342 – Read real sync data.....                                      | 614 |
| Table 343 – Read expected sync data .....                                 | 617 |
| Table 344 – Read PDev data .....                                          | 620 |
| Table 345 – Sync state info.....                                          | 625 |
| Table 346 – Write adjusted fiber optic data .....                         | 626 |
| Table 347 – Read real fiber optic data .....                              | 629 |
| Table 348 – Write MRP interface data.....                                 | 631 |
| Table 349 – Read MRP interface data.....                                  | 634 |
| Table 350 – Write MRP port data .....                                     | 636 |
| Table 351 – Read MRP port data .....                                      | 638 |
| Table 352 – Write FSU data.....                                           | 640 |
| Table 353 – Read FSU data.....                                            | 642 |
| Table 354 – Set time.....                                                 | 647 |
| Table 355 – Device Access.....                                            | 658 |
| Table 356 – Companion AR .....                                            | 658 |
| Table 357 – Media Redundancy.....                                         | 662 |
| Table 358 – Frame ID .....                                                | 663 |

|                                                                                  |     |
|----------------------------------------------------------------------------------|-----|
| Table 359 – Read AR data .....                                                   | 668 |
| Table 360 – State table application startup IO device.....                       | 673 |
| Table 361 – State table functions for startup IO device .....                    | 679 |
| Table 362 – State table startup manager IO device .....                          | 681 |
| Table 363 – State table functions for startup manager RT_CLASS_3 IO device ..... | 682 |
| Table 364 – State table IRT port state IO device (standard) .....                | 684 |
| Table 365 – State table functions for IRT port state IO device (standard).....   | 694 |
| Table 366 – State table IRT port state IO device (optimized).....                | 696 |
| Table 367 – State table neighborhood check.....                                  | 703 |
| Table 368 – State table functions for neighborhood check .....                   | 710 |
| Table 369 – State table PD parameter check IO device.....                        | 711 |
| Table 370 – State table functions PD parameter check IO device.....              | 716 |
| Table 371 – State table fiber optic maintenance required .....                   | 717 |
| Table 372 – State table fiber optic maintenance demanded.....                    | 718 |
| Table 373 – State table fiber optic diagnosis.....                               | 719 |
| Table 374 – State table for a submodule .....                                    | 721 |
| Table 375 – State table for plug .....                                           | 731 |
| Table 376 – State table for pull .....                                           | 732 |
| Table 377 – State table of PTCP behavior .....                                   | 734 |
| Table 378 – Functions used by PTCP behavior .....                                | 735 |
| Table 379 – State table IO controller during startup .....                       | 738 |
| Table 380 – State table startup manager IO controller .....                      | 741 |
| Table 381 – Functions for startup manager IO controller .....                    | 743 |

## INTRODUCTION

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

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

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

## INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

#### 1 Scope

##### 1.1 Overview

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

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

This standard defines in an abstract way the externally visible service provided by the type 10 fieldbus Application Layer in terms of

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

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

- 1) the FAL user at the boundary between the user and the Application Layer of the Fieldbus Reference Model, and
- 2) Systems Management at the boundary between the Application Layer and Systems Management of the Fieldbus Reference Model.

This standard specifies the structure and services of the type 10 IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI Application Layer Structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application-entities (AE) contained within the application processes. The FAL AE is composed of a set of object-oriented Application Service Elements (ASEs) and a Layer Management Entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for the management of the instances of FAL classes.

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the applications are not specified; only a definition of what requests and responses they can send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this standard to provide access to the FAL to control certain aspects of its operation.

## 1.2 Specifications

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

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

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

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

## 1.3 Conformance

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

There is conformance of equipment to this application layer service definition standard mainly achieved through implementation of the modeled behavior of an application layer user (e.g. see user state machines) accompanied by implementation of conforming application layer protocols that fulfill the application layer services as defined in this standard.

## 2 Normative references

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

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

IEC 61131-1, *Programmable controllers – Part 1: General information*

IEC 61131-3, *Programmable controllers – Part 3: Programming languages*

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

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

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

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

ISO/IEC TR 8802-1, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 1: Overview of Local Area Network Standards*

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

*Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

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

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

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

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

ISO/IEC 10646-1, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

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

IEEE 802-2001, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*

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

IEEE 802.1D-2004, *IEEE Standards for Local and Metropolitan Area Networks – Media Access Control (MAC) Bridges*

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

IEEE 802.3-2005, *IEEE Standards for Information technology – Telecommunications and information exchange between systems – Local and Metropolitan Area Networks – Specific Requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3 Terms, definitions, abbreviations, symbols and conventions

#### 3.1 Referenced terms and definitions

##### 3.1.1 ISO/IEC 7498-1 terms

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

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

##### 3.1.2 ISO/IEC 8822 terms

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

- a) abstract syntax
- b) presentation context

##### 3.1.3 ISO/IEC 9545 terms

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

- a) application-association
- b) application-context

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

#### 3.1.4 ISO/IEC 8824-1 terms

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

- a) object identifier
- b) type

### 3.2 Additional terms and definitions for distributed automation

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

#### 3.2.1

##### **active connection control object (ACCO)**

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

#### 3.2.2

##### **configuration data base**

interconnection information maintained by the ACCO ASE

#### 3.2.3

##### **connection**

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

#### 3.2.4

##### **connection channel**

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

#### 3.2.5

##### **consumer**

node or sink that is receiving data from a producer

#### 3.2.6

##### **consumerID**

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

#### 3.2.7

##### **data marshaling**

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

NOTE This is part of the abstract ORPC model.

### 3.2.8

#### **engineering**

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

### 3.2.9

#### **event**

instance of a change of conditions

### 3.2.10

#### **interface**

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

### 3.2.11

#### **interface definition language**

syntax and semantic of describing service parameters in a formal way

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

### 3.2.12

#### **interface pointer**

key attribute that unambiguously addresses an object interface instance

### 3.2.13

#### **logical device**

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

### 3.2.14

#### **method**

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

### 3.2.15

#### **object remote procedure call**

model for object oriented or component based remote method invocation

### 3.2.16

#### **physical device**

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

### 3.2.17

#### **property**

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

NOTE These services are generally named "get\_<Attribute Name>" or "set\_<Attribute Name>" and correspond with the IDL keywords "propget" and "propput" used in -5-10.

### 3.2.18

#### **provider**

source of a data connection.

### 3.2.19

#### **providerID**

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

**3.2.20****quality code**

additional status information of a data item

**3.2.21****quality code aware**

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

**3.2.22****quality code unaware**

opposite of quality code aware

**3.2.23****RT-Auto**

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

**3.2.24****runtime object model**

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

**3.3 Additional terms and definitions for decentralized periphery**

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

**3.3.1****alarm**

activation of an event that shows a critical state

**3.3.2****alarm ack**

acknowledgment of an event that shows a critical state

**3.3.3****alarm data object**

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

**3.3.4****allocate**

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

**3.3.5****application**

function or data structure for which data is consumed or produced

**3.3.6****application layer interoperability**

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

**3.3.7****application objects**

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

### 3.3.8

#### **application process**

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

### 3.3.9

#### **application process identifier**

distinguishes multiple application processes used in a device

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

### 3.3.10

#### **application process object**

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

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

### 3.3.11

#### **application process object class**

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

### 3.3.12

#### **application relationship**

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

### 3.3.13

#### **application relationship application service element**

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

### 3.3.14

#### **application relationship endpoint**

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

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

### 3.3.15

#### **attribute**

description of an externally visible characteristic or feature of an object

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

### 3.3.16

#### **backup**

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

### 3.3.17

#### **behavior**

indication of how an object responds to particular events

**3.3.18****channel**

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

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

**3.3.19****channel related diagnosis**

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

EXAMPLE open loop

**3.3.20****class**

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

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

**3.3.21****class attributes**

attribute that is shared by all objects within the same class

**3.3.22****class code**

unique identifier assigned to each object class

**3.3.23****class specific service**

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

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

**3.3.24****clear**

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

**3.3.25****client**

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

**3.3.26****common profile**

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

**3.3.27****communication data object**

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

**3.3.28****configuration check**

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

**3.3.29**

**configuration fault**

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

**3.3.30**

**configuration identifier**

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

**3.3.31**

**consume**

act of receiving data from a provider

**3.3.32**

**consumer**

node or sink receiving data from a provider

**3.3.33**

**context management**

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

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

**3.3.34**

**conveyance path**

unidirectional flow of APDUs across an application relationship

**3.3.35**

**cyclic**

repetitive in a regular manner

**3.3.36**

**data consistency**

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

**3.3.37**

**device**

physical hardware connected to the link

NOTE A device may contain more than one node.

**3.3.38**

**device ID**

vendor assigned device type identification

**3.3.39**

**device profile**

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

**3.3.40**

**diagnosis data object**

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

**3.3.41**

**diagnosis information**

all data available at the server for maintenance purposes

**3.3.42****dynamic reconfiguration**

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

**3.3.43****endpoint**

one of the communicating entities involved in a connection

**3.3.44****engineering**

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

**3.3.45****error**

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

**3.3.46****error class**

general grouping for related error definitions and corresponding error codes

**3.3.47****error code**

identification of a specific type of error within an error class

**3.3.48****event**

instance of a change of conditions

**3.3.49****extended channel related diagnosis**

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

EXAMPLE Link Fail

**3.3.50****fast startup**

accelerated procedure to setup an IO device after a power cycle

**3.3.51****frame**

unit of data transfer on a link

**3.3.52****identification data object**

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

**3.3.53****implicit AR endpoint**

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

**3.3.54****index**

address of a record data object within an application process

### **3.3.55**

#### **instance**

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

### **3.3.56**

#### **instance attributes**

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

### **3.3.57**

#### **instantiated**

object that has been created in a device

### **3.3.58**

#### **invocation**

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

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

### **3.3.59**

#### **IO controller**

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

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

### **3.3.60**

#### **IO data object**

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

### **3.3.61**

#### **IO device**

field device which acts as server for IO operation

### **3.3.62**

#### **IO parameter server**

server for application parameter of IO devices (client)

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

### **3.3.63**

#### **IO subsystem**

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

### **3.3.64**

#### **IO supervisor**

engineering device which manages commissioning and diagnosis of an IO system

### **3.3.65**

#### **IO system**

system composed of all its IO subsystems

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

### **3.3.66**

#### **Isochronous mode**

IO system operating tightly synchronized with a jitter of less than 1  $\mu$ s

**3.3.67****member**

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

**3.3.68****method**

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

**3.3.69****module**

hardware or logical component of a physical device

**3.3.70****network**

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

**3.3.71****object**

abstract representation of a particular component within a device, usually a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have a clearly defined interface and behavior

**3.3.72****object specific service**

service unique to the object class which defines it

**3.3.73****operate**

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

**3.3.74****peer**

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

**3.3.75****physical device**

automation or other network device

**3.3.76****point-to-point connection**

connection that exists between exactly two application objects

**3.3.77****primary**

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

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

**3.3.78****provider**

node or source sending data to one or many consumer

**3.3.79****PTCP domain**

certain number of PTCP subdomains in one IP subnet

**3.3.80**

**PTCP subdomain**

certain amount of DTEs with synchronized clocks

**3.3.81**

**record data object**

object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing and referenced by device/slot/subslot/index

**3.3.82**

**resource**

processing or information capability

**3.3.83**

**run**

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

**3.3.84**

**server**

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

**3.3.85**

**service**

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

**3.3.86**

**slot**

address of a structural unit within an IO device

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

**3.3.87**

**stop**

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

**3.3.88**

**submodule**

hardware or logical component of a module

**3.3.89**

**subslot**

address of a structural unit within a slot

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

**3.3.90**

**vendor ID**

central administrative number used as manufacturer identification

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

### 3.4 Additional terms and definitions for media redundancy

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

#### 3.4.1

##### **failure**

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

NOTE 1 After failure the item has a fault.

NOTE 2 "Failure" is an event, as distinguished from "fault", which is a state.

NOTE 3 This concept as defined does not apply to items considering of software only.

#### 3.4.2

##### **fault**

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

NOTE A fault is often the result of a failure of the item itself, but may exist without prior failure.

#### 3.4.3

##### **recovery**

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

#### 3.4.4

##### **recovery time**

time period required for recovery

#### 3.4.5

##### **redundancy**

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

NOTE In this context, the existence of more than one path (consisting of links and switches) between end nodes.

#### 3.4.6

##### **ring**

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

NOTE 1 Nodes are connected to one another in the logical shape of a circle.

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

#### 3.4.7

##### **ring port**

port of a switch to which a ring link is attached

#### 3.4.8

##### **ring link**

link that connects two switches of a ring

#### 3.4.9

##### **switch node**

MAC bridge as defined in IEEE 802.1D

NOTE It will be called "switch" in this context.

### 3.5 Abbreviations and symbols

|        |                                                    |
|--------|----------------------------------------------------|
| ACCO   | Active Connection Control Object                   |
| AE     | Application Entity                                 |
| AL     | Application Layer                                  |
| ALME   | Application Layer Management Entity                |
| ALP    | Application Layer Protocol                         |
| APO    | Application Object                                 |
| AP     | Application Process                                |
| APDU   | Application Protocol Data Unit                     |
| API    | Application Process Identifier                     |
| AR     | Application Relationship                           |
| AREP   | Application Relationship End Point                 |
| ASCII  | American Standard code for Information Interchange |
| ASE    | Application Service Element                        |
| CID    | Connection ID                                      |
| CIM    | Computer Integrated Manufacturing                  |
| CIP    | Control and Information Protocol                   |
| CM_API | Actual Packet Interval                             |
| CM_RPI | Requested Packet Interval                          |
| Cnf    | Confirmation                                       |
| COR    | Connection originator                              |
| CR     | Communication Relationship                         |
| CREP   | Communication Relationship End Point               |
| DL-    | (as a prefix) Data Link-                           |
| DLC    | Data Link Connection                               |
| DLL    | Data Link Layer                                    |
| DLM    | Data Link-management                               |
| DLSAP  | Data Link Service Access Point                     |
| DLSDU  | DL-service-data-unit                               |
| DNS    | Domain Name Service                                |
| DP     | Decentralised Peripherals                          |
| FAL    | Fieldbus Application Layer                         |
| FIFO   | First In First Out                                 |
| HMI    | Human-machine Interface                            |
| ID     | Identifier                                         |
| IDL    | Interface Definition Language                      |
| IEC    | International Electrotechnical Commission          |
| Ind    | Indication                                         |
| IP     | Internet Protocol                                  |
| ISO    | International Organization for Standardization     |
| LDev   | Logical device                                     |
| LME    | Layer Management Entity                            |
| O2T    | Originator to target (connection characteristics)  |
| OPT    | Originator to target (connection characteristics)  |
| ORPC   | Object Remote Procedure Call                       |
| OSI    | Open Systems Interconnect                          |
| PDev   | Physical device                                    |
| PDU    | Protocol Data Unit                                 |
| PL     | Physical Layer                                     |
| QoS    | Quality of Service                                 |
| QC     | Quality code                                       |
| REP    | Route Endpoint                                     |
| Req    | Request                                            |
| Rsp    | Response                                           |
| RT     | Runtime                                            |

|       |                                                             |
|-------|-------------------------------------------------------------|
| SAP   | Service Access Point                                        |
| SCL   | Security Level                                              |
| SDU   | Service Data Unit                                           |
| SEM   | State event matrix                                          |
| SMIB  | System Management Information Base                          |
| SMK   | System Management Kernel                                    |
| STD   | State transition diagram, used to describe object behaviour |
| S-VFD | Simple Virtual Field Device                                 |
| T2O   | Target to originator (connection characteristics)           |
| TPO   | Target to originator (connection characteristics)           |
| VAO   | Variable Object                                             |

### 3.6 Additional abbreviations and symbols for distributed automation

|      |                                  |
|------|----------------------------------|
| ACCO | Active Connection Control Object |
| IDL  | Interface Definition Language    |
| IP   | Internet Protocol                |
| LDev | Logical device                   |
| ORPC | Object Remote Procedure Call     |
| PDev | Physical device                  |
| QoS  | Quality of Service               |
| QC   | Quality code                     |
| RT   | Runtime                          |
| TCP  | Transmission Control Protocol    |

### 3.7 Additional abbreviations and symbols for decentralized periphery

|        |                                                    |
|--------|----------------------------------------------------|
| AE     | Application Entity                                 |
| AL     | Application Layer                                  |
| ALME   | Application Layer Management Entity                |
| ALP    | Application Layer Protocol                         |
| ALPMI  | Alarm Protocol machine Initiator                   |
| ALPMR  | Alarm Protocol machine Responder                   |
| AP     | Application Process                                |
| APDU   | Application Protocol Data Unit                     |
| API    | Application Process Identifier                     |
| APO    | Application Object                                 |
| AR     | Application Relationship                           |
| AREP   | Application Relationship End Point                 |
| ARP    | Address Resolution Protocol                        |
| ASCII  | American Standard code for Information Interchange |
| ASE    | Application Service Element                        |
| BMC    | Best Master Clock                                  |
| CM     | Context Management                                 |
| Cnf    | Confirmation                                       |
| CR     | Communication Relationship                         |
| CREP   | Communication Relationship End Point               |
| DCE    | OSF Distributed Computing Environment              |
| DCP    | Discovery and basic Configuration Protocol         |
| DCPMCR | DCP Multicast Receiver                             |
| DCPMCS | DCP Multicast Sender                               |
| DCPUCR | DCP Unicast Receiver                               |
| DCPUCS | DCP Unicast Sender                                 |
| DHCP   | Dynamic Host Configuration Protocol                |
| DIM    | Device Interface Module                            |
| DL-    | (as a prefix) Data Link-                           |
| DLC    | Data Link Connection                               |
| DLL    | Data Link Layer                                    |
| DLPDU  | Data Link-Protocol Data Unit                       |
| DLSDU  | DL-service-data-unit                               |
| DNS    | Domain Name Service                                |
| DTE    | Date Terminal Equipment                            |

|       |                                                                                   |
|-------|-----------------------------------------------------------------------------------|
| FAL   | Fieldbus Application Layer                                                        |
| FIFO  | First In First Out                                                                |
| GSDML | Generic Station Description Markup Language                                       |
| I&M   | Identification and Maintenance Profile                                            |
| IANA  | Internet Assigned Numbers Authority                                               |
| ICMP  | Internet Control Message Protocol                                                 |
| ID    | Identifier                                                                        |
| IEC   | International Electrotechnical Commission                                         |
| Ind   | Indication                                                                        |
| IOCS  | Input Output Object Consumer Status                                               |
| IOPS  | Input Output Object Provider Status                                               |
| IP    | Internet Protocol                                                                 |
| IR    | Isochronous Relay                                                                 |
| IRT   | Isochronous Real Time Protocol                                                    |
| ISO   | International Organization for Standardization                                    |
| IsoM  | Isochronous Mode                                                                  |
| LED   | Light Emitting Diode                                                              |
| LLDP  | Link Layer Discovery Protocol                                                     |
| LME   | Layer Management Entity                                                           |
| lsb   | Least Significant Bit                                                             |
| LT    | Length/type                                                                       |
| MAC   | Medium Access Control                                                             |
| msb   | Most Significant Bit                                                              |
| NCA   | Network Computing Architecture                                                    |
| OSI   | Open Systems Interconnect                                                         |
| PDU   | Protocol Data Unit                                                                |
| PI    | PROFIBUS International see <a href="http://www.profibus.com">www.profibus.com</a> |
| PL    | Physical Layer                                                                    |
| PTCP  | Precision Transparent Clock Protocol                                              |
| QoS   | Quality of Service                                                                |
| Req   | Request                                                                           |
| RPC   | Remote Procedure Call                                                             |
| Rsp   | Response                                                                          |
| RT    | Real Time Protocol                                                                |
| RTA   | Real Time Protocol Acyclic                                                        |
| RTC   | Real Time Protocol Cyclic                                                         |
| RTE   | Real Time Ethernet                                                                |
| SDU   | Service Data Unit                                                                 |
| TLV   | type Length Value (coding rule)                                                   |
| UDP   | User Datagram Protocol                                                            |
| UUID  | Universal Unique Identifier                                                       |
| VLAN  | Virtual Local Area Network                                                        |

### 3.8 Additional abbreviations and symbols for media redundancy

|     |                           |
|-----|---------------------------|
| MRC | Media Redundancy Client   |
| MRM | Media Redundancy Manager  |
| MRP | Media Redundancy Protocol |

### 3.9 Conventions

#### 3.9.1 Overview

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of two parts, its class specification, and its service specification.

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

### 3.9.2 General conventions

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

### 3.9.3 Conventions for class definitions

Class definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is shown below:

|                      |                                       |
|----------------------|---------------------------------------|
| <b>FAL ASE:</b>      | <b>ASE Name</b>                       |
| <b>CLASS:</b>        | <b>Class Name</b>                     |
| <b>CLASS ID:</b>     | <b>#</b>                              |
| <b>PARENT CLASS:</b> | <b>Parent Class Name</b>              |
| <b>ATTRIBUTES:</b>   |                                       |
| 1                    | (o) Key Attribute: numeric identifier |
| 2                    | (o) Key Attribute: name               |
| 3                    | (m) Attribute: attribute name(values) |
| 4                    | (m) Attribute: attribute name(values) |
| 4.1                  | (s) Attribute: attribute name(values) |
| 4.2                  | (s) Attribute: attribute name(values) |
| 4.3                  | (s) Attribute: attribute name(values) |
| 5.                   | (c) Constraint: constraint expression |
| 5.1                  | (m) Attribute: attribute name(values) |
| 5.2                  | (o) Attribute: attribute name(values) |
| 6                    | (m) Attribute: attribute name(values) |
| 6.1                  | (s) Attribute: attribute name(values) |
| 6.2                  | (s) Attribute: attribute name(values) |
| <b>SERVICES:</b>     |                                       |
| 1                    | (o) OpsService: service name          |
| 2.                   | (c) Constraint: constraint expression |
| 2.1                  | (o) OpsService: service name          |
| 3                    | (m) MgtService: service name          |

- (1) The "FAL ASE:" entry is the name of the FAL ASE that provides the services for the class being specified.
- (2) The "CLASS:" entry is the name of the class being specified. All objects defined using this template will be an instance of this class. The class may be specified by this standard, or by a user of this standard.
- (3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the FAL ASE that will provide the services for this class. When qualified by the identity of its FAL ASE, it unambiguously identifies the class within the scope of the FAL. The value "NULL" indicates that the class cannot be instantiated. Class IDs between 1 and 255 are reserved by this standard to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2048 are allocated for identifying user defined classes.
- (4) The "PARENT CLASS:" entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited for the class being defined, and therefore do not have to be redefined in the template for this class.

NOTE The parent-class "TOP" indicates that the class being defined is an initial class definition. The parent class TOP is used as a starting point from which all other classes are defined. The use of TOP is reserved for classes defined by this standard.

- (5) The "ATTRIBUTES" label indicate that the following entries are attributes defined for the class.
  - a) Each of the attribute entries contains a line number in column 1, a mandatory (m) / optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label

in column 3, a name or a conditional expression in column 4, and optionally a list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.

- b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.
- c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting is used to specify
  - i) fields of a structured attribute (4.1, 4.2, 4.3),
  - ii) attributes conditional on a constraint statement (5). Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2).
  - iii) the selection fields of a choice type attribute (6.1 and 6.2).

(6) The "SERVICES" label indicates that the following entries are services defined for the class.

- a) An (m) in column 2 indicates that the service is mandatory for the class, while an (o) indicates that it is optional. A (c) in this column indicates that the service is conditional. When all services defined for a class are defined as optional, at least one has to be selected when an instance of the class is defined.
- b) The label "OpsService" designates an operational service (1).
- c) The label "MgtService" designates a management service (2).
- d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

### 3.9.4 Conventions for service definitions

#### 3.9.4.1 General

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

#### 3.9.4.2 Service parameters

Service primitives are used to represent service user/service provider interactions (ISO/IEC 10731). They convey parameters which indicate information available in the user/provider interaction. In any particular interface, not all parameters need be explicitly stated.

The service specifications of this standard use a tabular format to describe the component parameters of the ASE service primitives. The parameters which apply to each group of service primitives are set out in tables. Each table consists of up to five columns for the

- 1) Parameter name,
- 2) request primitive,
- 3) indication primitive,
- 4) response primitive, and
- 5) confirm primitive.

One parameter (or component of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the column:

- M parameter is mandatory for the primitive
- U parameter is a User option, and may or may not be provided depending on dynamic usage of the service user. When not provided, a default value for the parameter is assumed.

- C parameter is conditional upon other parameters or upon the environment of the service user.
- (blank) parameter is never present.
- S parameter is a selected item.

Some entries are further qualified by items in brackets. These may be

- a) a parameter-specific constraint:
  - "(=)" indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.
- b) an indication that some note applies to the entry:
  - "(n)" indicates that the following note "n" contains additional information pertaining to the parameter and its use.

### 3.9.4.3 Service procedures

The procedures are defined in terms of

- the interactions between application entities through the exchange of fieldbus Application Protocol Data Units, and
- the interactions between an application layer service provider and an application layer service user in the same system through the invocation of application layer service primitives.

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

## 4 Concepts

NOTE Clause 9 of IEC/TR 61158-1:2007 describes the concepts of the application layer service descriptions and the templates used in this standard.

## 5 Data type ASE

### 5.1 General

#### 5.1.1 Overview

Fieldbus data types specify the machine independent syntax for application data conveyed by FAL services. The fieldbus application layer supports the definition and transfer of both basic and constructed data types. Encoding rules for the data types specified in this clause are provided in IEC 61158-6-10.

Basic types are atomic types that cannot be decomposed into more elemental types. Constructed types are types composed of basic types and other constructed types. Their complexity and depth of nesting is not constrained by this standard.

Data types are defined as instances of the data type class, as shown in Figure 1. Only a subset of the data types defined in this Clause are shown in this figure. Defining new types is accomplished by providing a numeric id and supplying values for the attributes defined for the data type class.

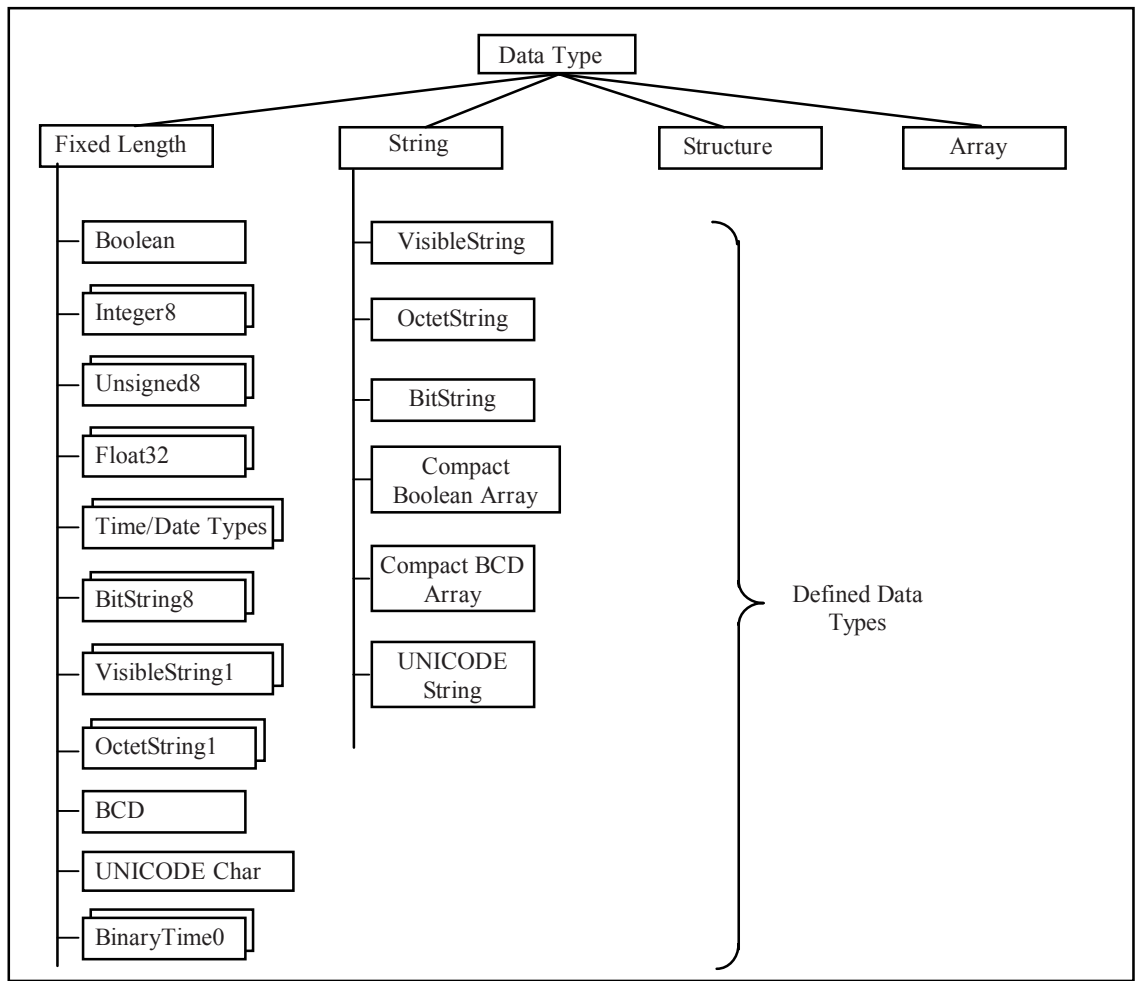


Figure 1 – Data type class hierarchy example

The data type definitions in Figure 1 are represented as a class/format/instance structure beginning with data type class entitled "Data type". The formats for data types are defined by the data type class and are represented in Figure 1.

The basic data classes are used to define fixed length and bitstring data types. Standard types taken from ISO/IEC 8824-1 are referred to as *simple* data types. Other standard basic data types are defined specifically for fieldbus applications and are referred to as *specific types*.

The constructed types specified in this standard are strings, arrays and structures. There are no standard types defined for arrays and structures.

### 5.1.2 Basic type overview

Most basic types are defined from a set of ISO/IEC 8824-1 types (simple types). Some ISO/IEC 8824-1 types have been extended for fieldbus specific use (specific types).

Simple types are ISO/IEC 8824-1 universal types. They are defined in this standard to provide them with fieldbus class identifiers.

Specific types are basic types defined specifically for use in the fieldbus environment. They are defined as simple class subtypes.

Basic types have a constant length. Two variations are defined, one for defining data types whose length is an integral number of octets, and one for defining data types whose length is bits.

NOTE Boolean, Integer, OctetString, VisibleString, and UniversalTime are defined in this standard for the purpose of assigning fieldbus class identifiers to them. This standard does not change their definitions as specified in ISO/IEC 8824-1.

### 5.1.3 Fixed length type overview

The length of Fixed length types is an integral number of octets.

### 5.1.4 Constructed type overview

#### 5.1.4.1 Strings

A string is composed of an ordered set, variable in number, of homogeneously typed fixed-length elements.

#### 5.1.4.2 Arrays

An array is composed of an ordered set of homogeneously typed elements. This standard places no restriction on the data type of array elements, but it does require that each element be of the same type. Once defined, the number of elements in an array may not be changed.

#### 5.1.4.3 Structures

A structure is made of an ordered set of heterogeneously typed elements called fields. Like arrays, this standard does not restrict the data type of fields. However, the fields within a structure do not have to be of the same type.

#### 5.1.4.4 Nesting level

This standard permits arrays and structures to contain arrays and structures. It places no restriction on the number of nesting levels allowed. However, the FAL services defined to access data provide for partial access to the level negotiated by the Initiate service. The default number of levels for partial access is one.

When an array or structure contains constructed elements, access to a single element in its entirety is always provided. Access to subelements of the constructed element is also provided, but only when explicitly negotiated during AR establishment, or when explicitly preconfigured on pre-established ARs.

NOTE For example, suppose that a data type named "employee" is defined to contain the structure "employee name", and "employee name" is defined to contain "last name" and "first name". To access the "employee" structure, the FAL permits independent access to the entire structure and to the first level field "employee name". Without explicitly negotiating partial access to more than one level, independent access to "last name" or "first name" would not be possible; their values could only be accessed together as a unit through access to "employee" or "employee name".

### 5.1.5 Specification of user defined data types

Users may find it necessary to define custom data types for their own applications. User defined types are supported by this standard as instances of data type classes.

User defined types are specified in the same manner that all FAL objects are specified. They are defined by providing values for the attributes specified for their class.

### 5.1.6 Transfer of user data

User data is transferred between applications by the FAL protocol. All encoding and decoding are performed by the FAL user.

The rules for encoding user data in FAL protocol data units is data type dependent. These rules are defined in IEC 61158-6-10. User-defined data types for which there are no encoding rules are transferred as a variable-length sequence of octets. The format of the data within the octet string is defined by the user.

## 5.2 Formal definition of data type objects

### 5.2.1 Data type class

#### 5.2.1.1 Template

The data type class specifies the root of the data type class tree. Its parent class "top" indicates the top of the FAL class tree.

|                      |                                                                 |
|----------------------|-----------------------------------------------------------------|
| <b>FAL ASE:</b>      | <b>DATA TYPE ASE</b>                                            |
| <b>CLASS:</b>        | <b>DATA TYPE</b>                                                |
| <b>CLASS ID:</b>     | <b>5 (FIXED LENGTH &amp; STRING), 6 (STRUCTURE), 12 (ARRAY)</b> |
| <b>PARENT CLASS:</b> | <b>TOP</b>                                                      |
| <b>ATTRIBUTES:</b>   |                                                                 |
| 1                    | (m) Key Attribute: Data type Numeric Identifier                 |
| 2                    | (o) Key Attribute: Data type Name                               |
| 3                    | (m) Attribute: Format (FIXED LENGTH, STRING, STRUCTURE, ARRAY)  |
| 4                    | (c) Constraint: Format = FIXED LENGTH   STRING                  |
| 4.1                  | (m) Attribute: Octet Length                                     |
| 5                    | (c) Constraint: Format = STRUCTURE                              |
| 5.1                  | (m) Attribute: Number of Fields                                 |
| 5.2                  | (m) Attribute: List of Fields                                   |
| 5.2.1                | (o) Attribute: Field Name                                       |
| 5.2.2                | (m) Attribute: Field Data type                                  |
| 6                    | (c) Constraint: Format = ARRAY                                  |
| 6.1                  | (m) Attribute: Number of Array Elements                         |
| 6.2                  | (m) Attribute: Array Element Data type                          |

#### 5.2.1.2 Attributes

##### Data type Numeric Identifier

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

##### Data type Name

This optional attribute identifies the name of the related data type.

##### Format

This attribute identifies the data type as a fixed-length, string, array, or data structure.

##### Octet Length

This conditional attribute defines the representation of the dimensions of the associated type object. It is present when the value of the format attribute is "FIXED LENGTH" or "STRING". For FIXED LENGTH data types, it represents the length in octets. For STRING data types, it represents the length in octets for a single element of a string.

##### Number of Fields

This conditional attribute defines the number of fields in a structure. It is present when the value of the format attribute is "STRUCTURE".

##### List of Fields

This conditional attribute is an ordered list of fields contained in the structure. Each field is specified by its number and its type. Fields are numbered sequentially from 0 (zero) in the order in which they occur. Partial access to fields within a structure is supported by identifying the field by number. This attribute is present when the value of the format attribute is "STRUCTURE".

**Field Name**

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

**Field Data type**

This conditional attribute specifies the data type of the field. It is present when the value of the format attribute is "STRUCTURE". This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric id, or by embedding a constructed data type definition here. When embedding a description, the Embedded Data type description shown below is used.

**Number of Array Elements**

This conditional attribute defines the number of elements for the array type. Array elements are indexed starting at "0" through "n-1" where the size of the array is "n" elements. This attribute is present when the value of the format attribute is "ARRAY".

**Array Element Data type**

This conditional attribute specifies the data type for the elements of an array. All elements of the array have the same data type. It is present when the value of the format attribute is "ARRAY". This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric id, or by embedding a constructed data type definition here. When embedding a description, the Embedded Data type description shown below is used.

**Embedded Data type Description**

This attribute is used to recursively define embedded data types within a structure or array. The template below defines its contents. The attributes shown in the template are defined above in the data type class, except for the Embedded Data type attribute, which is a recursive reference to this attribute. It is used to define nested elements.

**ATTRIBUTES:**

|       |                 |                                                |
|-------|-----------------|------------------------------------------------|
| 1     | (m) Attribute:  | Format(FIXED LENGTH, STRING, STRUCTURE, ARRAY) |
| 2     | (c) Constraint: | Format = FIXED LENGTH   STRING                 |
| 2.1   | (m) Attribute:  | Data type Numeric ID value                     |
| 2.2   | (m) Attribute:  | Octet Length                                   |
| 3     | (c) Constraint: | Format = STRUCTURE                             |
| 3.1   | (m) Attribute:  | Number of Fields                               |
| 3.2   | (m) Attribute:  | List of Fields                                 |
| 3.2.1 | (m) Attribute:  | Embedded Data type Description                 |
| 4     | (c) Constraint: | Format = ARRAY                                 |
| 4.1   | (m) Attribute:  | Number of Array Elements                       |
| 4.2   | (m) Attribute:  | Embedded Data type Description                 |

**5.3 FAL defined data types****5.3.1 Fixed length types****5.3.1.1 Boolean types****5.3.1.1.1 Boolean**

**CLASS:** Data type

**ATTRIBUTES:**

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

This data type expresses a Boolean data type with the values TRUE and FALSE.

**5.3.1.1.2 VARIANT\_BOOL**

**CLASS:** Data type

**ATTRIBUTES:**

|     |                              |   |              |
|-----|------------------------------|---|--------------|
| 1   | Data type Numeric Identifier | = | 2049         |
| 2   | Data type Name               | = | VARIANT_BOOL |
| 3   | Format                       | = | FIXED LENGTH |
| 4.1 | Octet Length                 | = | 2            |

This data type expresses a Boolean data type with the values TRUE (-1) and FALSE (0) (see Integer16).

### 5.3.1.2 Bitstring types

### 5.3.1.3 Currency types

### 5.3.1.4 Date types

#### 5.3.1.4.1 date

This data type is the same as Float64.

The data type date has a resolution in the range of one nanosecond. It is valid for dates between January 1st, 100, and December 31st, 9999. The value of 0.0 has been defined for December 30, 1899, 0:00. The integer part of the value represents the days after December 30th, 1899 (for dates before this day, the corresponding value is negative); the fractional part defines the time at that day.

NOTE Independent of the data type range of date, the devices may support for connections only the data type range of "January 1, 1900, 0:00" and greater.

#### 5.3.1.4.2 Date

**CLASS:** Data type

**ATTRIBUTES:**

|     |                              |   |              |
|-----|------------------------------|---|--------------|
| 1   | Data type Numeric Identifier | = | 50           |
| 2   | Data type Name               | = | Date         |
| 3   | Format                       | = | FIXED LENGTH |
| 4.1 | Octet Length                 | = | 7            |

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours with the most significant bit indicating Standard Time or Daylight Saving Time. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned8 data type and gives the year. The values 0 ... 50 correspond to the years 2000 to 2050, the values 51 ... 99 correspond to the years 1951 to 1999.

#### 5.3.1.4.3 TimeOfDay

**CLASS:** Data type

**ATTRIBUTES:**

|     |                              |   |              |
|-----|------------------------------|---|--------------|
| 1   | Data type Numeric Identifier | = | 12           |
| 2   | Data type Name               | = | TimeOfDay    |
| 3   | Format                       | = | FIXED LENGTH |
| 4.1 | Octet Length                 | = | 6            |

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date counting the days from January 1, 1984.

**5.3.1.4.4 TimeOfDay with date indication**

This data type is the same as the TimeOfDay data type defined above.

**5.3.1.4.5 TimeOfDay without date indication**

**CLASS:** Data type

**ATTRIBUTES:**

- |     |                              |                                     |
|-----|------------------------------|-------------------------------------|
| 1   | Data type Numeric Identifier | = 52                                |
| 2   | Data type Name               | = TimeOfDay without date indication |
| 3   | Format                       | = FIXED LENGTH                      |
| 4.1 | Octet Length                 | = 4                                 |

This data type is composed of one element of an unsigned value and expresses the time of day without date indication. The element is an Unsigned32 data type and gives the time after the midnight in milliseconds.

**5.3.1.4.6 TimeDifference**

**CLASS:** Data type

**ATTRIBUTES:**

- |     |                              |                  |
|-----|------------------------------|------------------|
| 1   | Data type Numeric Identifier | = 13             |
| 2   | Data type Name               | = TimeDifference |
| 3   | Format                       | = FIXED LENGTH   |
| 4.1 | Octet Length                 | = 4 or 6         |

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The optional second element is an Unsigned16 data type that provides the difference in days.

**5.3.1.4.7 TimeDifference with date indication**

**CLASS:** Data type

**ATTRIBUTES:**

- |     |                              |                                       |
|-----|------------------------------|---------------------------------------|
| 1   | Data type Numeric Identifier | = 53                                  |
| 2   | Data type Name               | = TimeDifference with date indication |
| 3   | Format                       | = FIXED LENGTH                        |
| 4.1 | Octet Length                 | = 6                                   |

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The second element is an Unsigned16 data type that provides the difference in number of days.

**5.3.1.4.8 TimeDifference without date indication**

**CLASS:** Data type

**ATTRIBUTES:**

- |     |                              |                                          |
|-----|------------------------------|------------------------------------------|
| 1   | Data type Numeric Identifier | = 54                                     |
| 2   | Data type Name               | = TimeDifference without date indication |
| 3   | Format                       | = FIXED LENGTH                           |
| 4.1 | Octet Length                 | = 4                                      |

This data type is composed of one element of an unsigned value that express the difference in time. The element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds.

### 5.3.1.5 Enumerated types

#### 5.3.1.5.1 PERSISTDEF

**CLASS:** Data type

**ATTRIBUTES:**

- 1 Data type Numeric Identifier = 2050
- 2 Data type Name = PERSISTDEF
- 3 Format = FIXED LENGTH
- 4.1 Octet Length = 2

The allowed values are shown in Table 1.

**Table 1 – PERSISTDEF**

| Value                |
|----------------------|
| CBAVolatile          |
| CBAPendingPersistent |
| CBAPersistent        |

#### 5.3.1.5.2 VARTYPE

**CLASS:** Data type

**ATTRIBUTES:**

- 1 Data type Numeric Identifier = 2051
- 2 Data type Name = VARTYPE
- 3 Format = FIXED LENGTH
- 4.1 Octet Length = 2

The VARTYPE specifies the data type that governs the interpretation of the data. The allowed values are shown in Table 2.

Table 2 – VARTYPE

| Value                                                                       | Data type                                  |
|-----------------------------------------------------------------------------|--------------------------------------------|
| VT_EMPTY                                                                    | no value                                   |
| VT_NULL                                                                     | null value (no valid data)                 |
| VT_BOOL                                                                     | VARIANT_BOOL                               |
| VT_I1                                                                       | char                                       |
| VT_I2                                                                       | short                                      |
| VT_I4                                                                       | long                                       |
| VT_UI1                                                                      | unsigned char                              |
| VT_UI2                                                                      | unsigned short                             |
| VT_UI4                                                                      | unsigned long                              |
| VT_R4                                                                       | float                                      |
| VT_R8                                                                       | double                                     |
| VT_DATE                                                                     | date                                       |
| VT_BSTR                                                                     | BSTR                                       |
| VT_ERROR                                                                    | HRESULT                                    |
| VT_SAFEARRAY_BOOL                                                           | SAFEARRAY (VARIANT_BOOL)                   |
| VT_SAFEARRAY_I1                                                             | SAFEARRAY (char)                           |
| VT_SAFEARRAY_I2                                                             | SAFEARRAY (short)                          |
| VT_SAFEARRAY_I4                                                             | SAFEARRAY (long)                           |
| VT_SAFEARRAY_UI1                                                            | SAFEARRAY (unsigned char)                  |
| VT_SAFEARRAY_UI2                                                            | SAFEARRAY (unsigned short)                 |
| VT_SAFEARRAY_UI4                                                            | SAFEARRAY (unsigned long)                  |
| VT_SAFEARRAY_R4                                                             | SAFEARRAY (float)                          |
| VT_SAFEARRAY_R8                                                             | SAFEARRAY (double)                         |
| VT_SAFEARRAY_DATE                                                           | SAFEARRAY (date)                           |
| VT_SAFEARRAY_BSTR                                                           | SAFEARRAY (BSTR)                           |
| VT_SAFEARRAY_ERROR                                                          | SAFEARRAY (HRESULT)                        |
| VT_USERDEFINED                                                              | userdefined struct                         |
| VT_DISPATCH                                                                 | Interface Pointer to an Dispatch interface |
| VT_UNKNOWN                                                                  | Interface Pointer to an Unknown interface  |
| NOTE All defined structured data types should use the value VT_USERDEFINED. |                                            |

### 5.3.1.5.3 ITEMQUALITYDEF

**CLASS:** Data type

**ATTRIBUTES:**

- |     |                                |                |
|-----|--------------------------------|----------------|
| 1   | Data type Numeric Identifier = | 2052           |
| 2   | Data type Name =               | ITEMQUALITYDEF |
| 3   | Format =                       | FIXED LENGTH   |
| 4.1 | Octet Length =                 | 1              |

This data type contains the status information of the related data. It consists of three portions: Quality, Substatus and Limits. There are four states of quality (Bad - the value is not useful; Uncertain - the quality of the value is less than normal, but the value may still be useful; Good (Non Cascade) - the quality of the value is good, possible alarm conditions may be indicated by the substatus; Good (Cascade) - the value may be used in control), a set of sub-status values for each quality, and four states of the limits (OK - the value is free to move; low limited (LL) - the value has acceded its low limits, high limited (HL) - the value has acceded its high limits; constant (C) - the value cannot move, no matter what the process does). The allowed values are shown in Table 3.

Table 3 – ITEMQUALITYDEF

| Quality | Value                 | Description                                                             |
|---------|-----------------------|-------------------------------------------------------------------------|
| Bad     | BadNonSpecific        | There is no specific reason why the value is bad. Used for propagation. |
|         | BadNonSpecificLL      | and low limited                                                         |
|         | BadNonSpecificHL      | and high limited                                                        |
|         | BadNonSpecificC       | and constant                                                            |
|         | BadConfigurationError | Set if the value is not useful because there is some other              |

| Quality   | Value                      | Description                                                                                                                                                                                                                                |
|-----------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |                            | problem with the block, depending on what a specific producer can detect.                                                                                                                                                                  |
|           | BadConfigurationErrorLL    | and low limited                                                                                                                                                                                                                            |
|           | BadConfigurationErrorHL    | and high limited                                                                                                                                                                                                                           |
|           | BadConfigurationErrorC     | and constant                                                                                                                                                                                                                               |
|           | BadNotConnected            | Set if this input is required to be connected and is not connected.                                                                                                                                                                        |
|           | BadNotConnectedLL          | and low limited                                                                                                                                                                                                                            |
|           | BadNotConnectedHL          | and high limited                                                                                                                                                                                                                           |
|           | BadNotConnectedC           | and constant                                                                                                                                                                                                                               |
|           | BadDeviceFailure           | Set if the source of the value is affected by a device failure.                                                                                                                                                                            |
|           | BadDeviceFailureLL         | and low limited                                                                                                                                                                                                                            |
|           | BadDeviceFailureHL         | and high limited                                                                                                                                                                                                                           |
|           | BadDeviceFailureC          | and constant                                                                                                                                                                                                                               |
|           | BadSensorFailure           | Set if the device can determine this condition. The limits define which direction has been exceeded.                                                                                                                                       |
|           | BadSensorFailureLL         | and low limited                                                                                                                                                                                                                            |
|           | BadSensorFailureHL         | and high limited                                                                                                                                                                                                                           |
|           | BadSensorFailureC          | and constant                                                                                                                                                                                                                               |
|           | BadLastKnownValue          | Set if this value had been set by communication, which has now failed.                                                                                                                                                                     |
|           | BadLastKnownValueLL        | and low limited                                                                                                                                                                                                                            |
|           | BadLastKnownValueHL        | and high limited                                                                                                                                                                                                                           |
|           | BadLastKnownValueC         | and constant                                                                                                                                                                                                                               |
|           | BaORPCmFailure             | Set if there has never been any communication with this value since it was last Out of Service.                                                                                                                                            |
|           | BaORPCmFailureLL           | and low limited                                                                                                                                                                                                                            |
|           | BaORPCmFailureHL           | and high limited                                                                                                                                                                                                                           |
|           | BaORPCmFailureC            | and constant                                                                                                                                                                                                                               |
|           | BadOutOfService            | The value is not reliable because the block is not being evaluated, and may be under construction by a configuration tool. It is set if the block mode is O/S.                                                                             |
|           | BadOutOfServiceLL          | and low limited                                                                                                                                                                                                                            |
|           | BadOutOfServiceHL          | and high limited                                                                                                                                                                                                                           |
|           | BadOutOfServiceC           | and constant                                                                                                                                                                                                                               |
| Uncertain | UncertainNonSpecific       | There is no specific reason why the value is uncertain. Used for propagation.                                                                                                                                                              |
|           | UncertainNonSpecificLL     | and low limited                                                                                                                                                                                                                            |
|           | UncertainNonSpecificHL     | and high limited                                                                                                                                                                                                                           |
|           | UncertainNonSpecificC      | and constant                                                                                                                                                                                                                               |
|           | UncertainLastUsableValue   | Whatever was writing this value has stopped doing so. This is used for fail safe handling.                                                                                                                                                 |
|           | UncertainLastUsableValueLL | and low limited                                                                                                                                                                                                                            |
|           | UncertainLastUsableValueHL | and high limited                                                                                                                                                                                                                           |
|           | UncertainLastUsableValueC  | and constant                                                                                                                                                                                                                               |
|           | UncertainSubstituteSet     | Predefined value is used instead of the calculated one. This is used for fail safe handling.                                                                                                                                               |
|           | UncertainSubstituteSetLL   | and low limited                                                                                                                                                                                                                            |
|           | UncertainSubstituteSetHL   | and high limited                                                                                                                                                                                                                           |
|           | UncertainSubstituteSetC    | and constant                                                                                                                                                                                                                               |
|           | UncertainInitialValue      | Value of volatile parameters during and after the reset of the device or a parameter.                                                                                                                                                      |
|           | UncertainInitialValueLL    | and low limited                                                                                                                                                                                                                            |
|           | UncertainInitialValueHL    | and high limited                                                                                                                                                                                                                           |
|           | UncertainInitialValueC     | and constant                                                                                                                                                                                                                               |
|           | UncertainSensorNotAccurate | Set if the value is at one of the sensor limits. The limits define which direction has been exceeded. Also set if the device can determine that the sensor has reduced accuracy (e.g. degraded analyzer), in which case no limits are set. |

| Quality                | Value                               | Description                                                                                                                           |
|------------------------|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
|                        | UncertainSensorNotAccurateLL        | and low limited                                                                                                                       |
|                        | UncertainSensorNotAccurateHL        | and high limited                                                                                                                      |
|                        | UncertainSensorNotAccurateC         | and constant                                                                                                                          |
|                        | UncertainEngineeringUnitsExceeded   | Set if the value lies outside of the range of values defined for this parameter. The limits define which direction has been exceeded. |
|                        | UncertainEngineeringUnitsExceededLL | and low limited                                                                                                                       |
|                        | UncertainEngineeringUnitsExceededHL | and high limited                                                                                                                      |
|                        | UncertainEngineeringUnitsExceededC  | and constant                                                                                                                          |
|                        | UncertainSubNormal                  | Set if a value derived from multiple values has less than the required number of Good sources.                                        |
|                        | UncertainSubNormalLL                | and low limited                                                                                                                       |
|                        | UncertainSubNormalHL                | and high limited                                                                                                                      |
|                        | UncertainSubNormalC                 | and constant                                                                                                                          |
|                        | UncertainConfigurationError         | Set if there is some inconsistency regarding the parameterization or configuration, depending on what a specific producer can detect. |
|                        | UncertainConfigurationErrorLL       | and low limited                                                                                                                       |
|                        | UncertainConfigurationErrorHL       | and high limited                                                                                                                      |
|                        | UncertainConfigurationErrorC        | and constant                                                                                                                          |
|                        | UncertainSimulatedValue             | Set when the process value is written by the operator while the block is in manual mode.                                              |
|                        | UncertainSimulatedValueLL           | and low limited                                                                                                                       |
|                        | UncertainSimulatedValueHL           | and high limited                                                                                                                      |
|                        | UncertainSimulatedValueC            | and constant                                                                                                                          |
|                        | UncertainSensorCalibration          | Set during the active calibration process together with the current measured value.                                                   |
| Good<br>Non<br>Cascade | UncertainSensorCalibrationLL        | and low limited                                                                                                                       |
|                        | UncertainSensorCalibrationHL        | and high limited                                                                                                                      |
|                        | UncertainSensorCalibrationC         | and constant                                                                                                                          |
|                        | GoodNonCascOk                       | No error or special condition is associated with this value.                                                                          |
|                        | GoodNonCascOkC                      | and constant                                                                                                                          |
|                        | GoodNonCascActiveUpdateEvent        | Set if the value is good and the block has an active Update Event.                                                                    |
|                        | GoodNonCascActiveUpdateEventLL      | and low limited                                                                                                                       |
|                        | GoodNonCascActiveUpdateEventHL      | and high limited                                                                                                                      |
|                        | GoodNonCascActiveUpdateEventC       | and constant                                                                                                                          |
|                        | GoodNonCascActiveAdvisoryAlarm      | Set if the value is good and the block has an active Alarm with a priority less than 8.                                               |
|                        | GoodNonCascActiveAdvisoryAlarmLL    | and low limited                                                                                                                       |
|                        | GoodNonCascActiveAdvisoryAlarmHL    | and high limited                                                                                                                      |
|                        | GoodNonCascActiveAdvisoryAlarmC     | and constant                                                                                                                          |
|                        | GoodNonCascActiveCriticalAlarm      | Set if the value is good and the block has an active Alarm with a priority greater than or equal to 8.                                |
|                        | GoodNonCascActiveCriticalAlarmLL    | and low limited                                                                                                                       |
|                        | GoodNonCascActiveCriticalAlarmHL    | and high limited                                                                                                                      |
|                        | GoodNonCascActiveCriticalAlarmC     | and constant                                                                                                                          |
|                        | GoodNonCascUnackUpdateEvent         | Set if the value is good and the block has an unacknowledged Update Event.                                                            |
|                        | GoodNonCascUnackUpdateEventLL       | and low limited                                                                                                                       |
|                        | GoodNonCascUnackUpdateEventHL       | and high limited                                                                                                                      |
|                        | GoodNonCascUnackUpdateEventC        | and constant                                                                                                                          |
|                        | GoodNonCascUnackAdvisoryAlarm       | Set if the value is good and the block has an unacknowledged Alarm with a priority less than 8.                                       |
|                        | GoodNonCascUnackAdvisoryAlarmLL     | and low limited                                                                                                                       |
|                        | GoodNonCascUnackAdvisoryAlarmHL     | and high limited                                                                                                                      |
|                        | GoodNonCascUnackAdvisoryAlarmC      | and constant                                                                                                                          |
|                        | GoodNonCascUnackCriticalAlarm       | Set if the value is good and the block has an unacknowledged                                                                          |