

DIN EN 61158-5-5

ICS 35.100.70; 35.160

Ersatzvermerk
siehe unten**Industrielle Kommunikationsnetze –
Feldbusse –****Teil 5-5: Dienstfestlegungen des Application Layer
(Anwendungsschicht) – Typ 5-Elemente (IEC 61158-5-5:2007);
Englische Fassung EN 61158-5-5:2008, nur auf CD-ROM**

Industrial communication networks –

Fieldbus specifications –

Part 5-5: Application layer service definition – Type 5 elements (IEC 61158-5-5:2007);

English version EN 61158-5-5:2008, only on CD-ROM

Réseaux de communication industriels –

Spécifications des bus de terrain des couches d'application –

Partie 5-5: Définition des services – Éléments de type 5 (CEI 61158-5-5:2007);

Version anglaise EN 61158-5-5:2008, seulement en CD-ROM

Ersatzvermerk

Mit DIN EN 61158-5-2:2008-09, DIN EN 61158-5-3:2008-09, DIN EN 61158-5-4:2008-09,
DIN EN 61158-5-7:2008-09, DIN EN 61158-5-8:2008-09, DIN EN 61158-5-9:2008-09,
DIN EN 61158-5-10:2008-09, DIN EN 61158-5-11:2008-09, DIN EN 61158-5-12:2008-09,
DIN EN 61158-5-13:2008-09, DIN EN 61158-5-14:2008-09, DIN EN 61158-5-15:2008-09,
DIN EN 61158-5-16:2008-09, DIN EN 61158-5-17:2008-09, DIN EN 61158-5-18:2008-09,
DIN EN 61158-5-19:2008-09 und DIN EN 61158-5-20:2008-09 Ersatz für DIN EN 61158-5:2005-05
Siehe jedoch Beginn der Gültigkeit

Gesamtumfang 512 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Beginn der Gültigkeit

Die von CENELEC am 2008-02-01 angenommene EN 61158-5-5 gilt als DIN-Norm ab 2008-09-01.

Daneben darf DIN EN 61158-5:2005-05 noch bis 2011-02-01 angewendet werden.

Nationales Vorwort

Mit DIN EN 61158-5-2:2008-09, DIN EN 61158-5-3:2008-09, DIN EN 61158-5-4:2008-09, DIN EN 61158-5-7:2008-09, DIN EN 61158-5-8:2008-09, DIN EN 61158-5-9:2008-09, DIN EN 61158-5-10:2008-09, DIN EN 61158-5-11:2008-09, DIN EN 61158-5-12:2008-09, DIN EN 61158-5-13:2008-09, DIN EN 61158-5-14:2008-09, DIN EN 61158-5-15:2008-09, DIN EN 61158-5-16:2008-09, DIN EN 61158-5-17:2008-09, DIN EN 61158-5-18:2008-09, DIN EN 61158-5-19:2008-09 und DIN EN 61158-5-20:2008-09 Ersatz für DIN EN 61158-5:2005-05.

Vorausgegangener Norm-Entwurf: E DIN EN 61158-500:2006-11.

Für diese Norm ist das nationale Arbeitsgremium K 956 „Feldbus“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Die enthaltene IEC-Publikation wurde vom SC 65C „Industrial networks“ erarbeitet.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (maintenance result date) unverändert bleiben soll, das auf der IEC-Website unter „<http://webstore.iec.ch>“ zu dieser Publikation angegeben ist. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als Europäische Norm EN 60068 durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Einführung

Ein Feldbus ist ein digitaler serieller Datenbus für die Kommunikation zwischen Geräten der industriellen Automatisierungstechnik wie beispielsweise Messeinrichtungen, Stellgliedern, Reglern und speicherprogrammierbaren Steuerungen. Die Normen der Reihe DIN EN 61158 legen eine Anzahl von Feldbus-Protokolltypen fest.

Die vorliegende Norm legt die Dienste fest, die von einem Feldbus-Application Layer des Typs 5 (Fieldbus Foundation HSE) zur Verfügung gestellt werden.

Änderungen

Gegenüber DIN EN 61158-5:2005-05 wurden folgende Änderungen vorgenommen:

- a) Streichung des Typ 6-Feldbusses wegen zu geringer Marktbedeutung;
- b) Hinzufügung von weiteren Feldbustypen;
- c) Aufteilung in mehrere Teile, jeweils als 5-2, 5-3, ..., 5-20 benummert.

Frühere Ausgaben

DIN V 19245-1: 1988-01

DIN 19245-1: 1991-04

DIN 19245-2: 1991-04

DIN EN 50170-1: 1997-07

DIN EN 50170-2: 1997-07

DIN EN 50170-3: 1997-07

DIN EN 50254: 1999-07

DIN EN 50170/A1: 2002-11

DIN EN 50170/A2: 2000-08

DIN EN 50170/A3: 2002-11

DIN EN 61158-5: 2005-05

— Leerseite —

Deutsche Fassung

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

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

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

Diese Europäische Norm wurde von CENELEC am 2008-02-01 angenommen. Die CENELEC-Mitglieder sind gehalten, die CEN/CENELEC-Geschäftsordnung zu erfüllen, in der die Bedingungen festgelegt sind, unter denen dieser Europäischen Norm ohne jede Änderung der Status einer nationalen Norm zu geben ist.

Auf dem letzten Stand befindliche Listen dieser nationalen Normen mit ihren bibliographischen Angaben sind beim Zentralsekretariat oder bei jedem CENELEC-Mitglied auf Anfrage erhältlich.

Diese Europäische Norm besteht in drei offiziellen Fassungen (Deutsch, Englisch, Französisch). Eine Fassung in einer anderen Sprache, die von einem CENELEC-Mitglied in eigener Verantwortung durch Übersetzung in seine Landessprache gemacht und dem Zentralsekretariat mitgeteilt worden ist, hat den gleichen Status wie die offiziellen Fassungen.

CENELEC-Mitglieder sind die nationalen elektrotechnischen Komitees von Belgien, Bulgarien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Griechenland, Irland, Island, Italien, Lettland, Litauen, Luxemburg, Malta, den Niederlanden, Norwegen, Österreich, Polen, Portugal, Rumänien, Schweden, der Schweiz, der Slowakei, Slowenien, Spanien, der Tschechischen Republik, Ungarn, dem Vereinigten Königreich und Zypern.

CENELEC

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

Zentralsekretariat: rue de Stassart 35, B-1050 Brüssel

© 2008 CENELEC – Alle Rechte der Verwertung, gleich in welcher Form und in welchem Verfahren, sind weltweit den Mitgliedern von CENELEC vorbehalten.

Ref. Nr. EN 61158-5-5:2008 D

Vorwort

Der Text des Schriftstücks 65C/475/FDIS, zukünftige 1. Ausgabe von IEC 61158-5-5, ausgearbeitet von dem SC 65C „Industrial networks“ des IEC/TC 65 „Industrial-process measurement, control and automation“, wurde der IEC-CENELEC Parallelen Abstimmung unterworfen und von CENELEC am 2008-02-01 als EN 61158-5-5 angenommen.

Dieser und die weiteren Teile der Normenreihe EN 61158-5 ersetzen EN 61158-5:2004.

Gegenüber EN 61158-5:2004 wurden die folgenden Änderungen durchgeführt:

- Streichung des Typ 6-Feldbusses wegen zu geringer Marktbedeutung;
- Hinzufügung von weiteren Feldbustypen;
- Aufteilung in mehrere Teile, jeweils als 5-2, 5-3, ..., 5-20 benummert.

Nachstehende Daten wurden festgelegt:

- spätestes Datum, zu dem die EN auf nationaler Ebene durch Veröffentlichung einer identischen nationalen Norm oder durch Anerkennung übernommen werden muss (dop): 2008-11-01
- spätestes Datum, zu dem nationale Normen, die der EN entgegenstehen, zurückgezogen werden müssen (dow): 2011-02-01

ANMERKUNG Die Anwendung einiger der aufgeführten Protokolltypen wird durch die Inhaber der Rechte an geistigem Eigentum hieran eingeschränkt. Die Zusage zur begrenzten Freigabe ihrer Rechte an geistigem Eigentum durch die Inhaber erlaubt in jedem Fall die Anwendung eines bestimmten Data Link Layer-Protokolls zusammen mit einem Physical Layer- und einem Application Layer-Protokoll in Typzusammenstellungen, die in der Normenreihe EN 61784 angegeben sind. Die Anwendung der verschiedenen Protokolltypen in anderen Zusammenstellungen erfordert möglicherweise die Erlaubnis des jeweiligen Inhabers der Rechte an dem entsprechenden geistigen Eigentum.

Der Anhang ZA wurde von CENELEC hinzugefügt.

Anerkennungsnotiz

Der Text der Internationalen Norm IEC 61158-5-5:2007 wurde von CENELEC ohne irgendeine Abänderung als Europäische Norm angenommen.

In der offiziellen Fassung sind unter „Literaturhinweise“ zu den aufgelisteten Normen die nachstehenden Anmerkungen einzutragen:

- IEC 61158-3-1 ANMERKUNG Harmonisiert als EN 61158-3-1:2008 (nicht modifiziert).
- IEC 61158-4-1 ANMERKUNG Harmonisiert als EN 61158-4-1:2008 (nicht modifiziert).
- IEC 61158-6-5 ANMERKUNG Harmonisiert als EN 61158-6-5:2008 (nicht modifiziert).
- IEC 61784-1 ANMERKUNG Harmonisiert als EN 61784-1:2008 (nicht modifiziert).

CONTENTS

INTRODUCTION.....	11
1 Scope	12
1.1 Overview	12
1.2 Specifications	13
1.3 Conformance	13
2 Normative references	13
3 Terms and definitions	14
3.1 ISO/IEC 7498-1 terms.....	14
3.2 ISO/IEC 8822 terms.....	14
3.3 ISO/IEC 9545 terms.....	14
3.4 ISO/IEC 8824 terms.....	14
3.5 Fieldbus data-link layer terms	14
3.6 Fieldbus application layer specific definitions	15
3.7 Abbreviations and symbols	24
3.8 Conventions	26
4 Concepts.....	29
5 Data type ASE.....	29
5.1 Overview	29
5.2 Formal definition of data type objects	29
5.3 FAL defined data types	31
5.4 Data type ASE service specification	68
6 Communication model specification	68
6.1 Concepts.....	68
6.2 ASEs.....	68
6.3 ARs.....	209
6.4 Summary of FAL classes	232
6.5 Permitted FAL services by AREP role	234
7 Type 5 communication model specification.....	235
7.1 Concepts.....	235
7.2 ASEs.....	256
7.3 FDA sessions	292
7.4 Summary of FAL Type 9 and Type 5 classes	301
7.5 Permitted FAL Type 9 and Type 5 services by AREP role	302
8 Type 7 communication model specification.....	304
8.1 Concepts.....	304
8.2 ASEs.....	320
8.3 ARs.....	486
Bibliography	507
Annex ZA (normative) Normative references to international publications with their corresponding European publications.....	508
Figure 1 – The AR ASE conveys APDUs between APs	98
Figure 2 – 1-to-1 AR establishment	110
Figure 3 – 1-to-many AR establishment.....	110
Figure 4 – Event model overview.....	149

Figure 5 – Residence timeliness	223
Figure 6 – Synchronized timeliness	224
Figure 7 – Residence timeliness	229
Figure 8 – Synchronized timeliness	230
Figure 9 – VCR initiation	241
Figure 10 – Misordered message handling	247
Figure 11 – FF SM port message processing order	248
Figure 12 – FF FDA port message processing order	249
Figure 13 – FF TCP connection message processing order	249
Figure 14 – Session endpoint message processing order	249
Figure 15 – FDA LAN redundancy port message processing order	250
Figure 16 – Message processing by receiving entity	250
Figure 17 – Organisation of the ASEs and ARs	305
Figure 18 – Object model of the MPS ASE	324
Figure 19 – Time-out evaluation net	336
Figure 20 – Asynchronous promptness status evaluation net	340
Figure 21 – Synchronous promptness status evaluation net	341
Figure 22 – Punctual promptness status evaluation net	343
Figure 23 – Asynchronous refreshment status evaluation net	346
Figure 24 – Synchronous refreshment status evaluation net	347
Figure 25 – Punctual refreshment status evaluation net	349
Figure 26 – A_Readloc service procedure	351
Figure 27 – A_Writeloc service procedure	353
Figure 28 – A_Update service procedure	354
Figure 29 – A_Readfar service procedure	357
Figure 30 – A_Writefar service procedure	359
Figure 31 – A_Sent service procedure	360
Figure 32 – A_Received service procedure	361
Figure 33 – A_Read service procedure	363
Figure 34 – A_Read service state machine	363
Figure 35 – A_Write service procedure	365
Figure 36 – A_Write service state machine	366
Figure 37 – Model of a resynchronised variable	369
Figure 38 – Principles for resynchronisation of a produced variable	370
Figure 39 – Resynchronisation mechanism state machine for a produced variable	372
Figure 40 – Asynchronous refreshment private mechanism evaluation net	373
Figure 41 – Asynchronous refreshment public mechanism evaluation net	374
Figure 42 – Synchronous refreshment private mechanism evaluation net	375
Figure 43 – Synchronous refreshment public mechanism evaluation net	376
Figure 44 – Punctual refreshment private mechanism evaluation net	377
Figure 45 – Punctual refreshment public mechanism evaluation net	378
Figure 46 – Principles for the resynchronisation of a consumed variable	379
Figure 47 – Resynchronisation mechanism state machine for consumed variable	381

Figure 48 – Asynchronous promptness public mechanism evaluation net	382
Figure 49 – Asynchronous promptness private mechanism evaluation net	383
Figure 50 – Synchronous promptness public mechanism evaluation net	384
Figure 51 – Synchronous promptness private mechanism evaluation net	385
Figure 52 – Punctual promptness public mechanism evaluation net	387
Figure 53 – Punctual promptness private mechanism evaluation net	388
Figure 54 – Spatial consistency list variables interchange mechanism	390
Figure 55 – Spatial consistency – consistency variable interchange mechanism	391
Figure 56 – Spatial consistency – list recovery mechanism	391
Figure 57 – Spatial consistency – validity of the spatial consistency status	392
Figure 58 – Object model of a variable list	392
Figure 59 – A_Readlist service procedure	398
Figure 60 – Consistency variable value evaluation net	404
Figure 61 – Consistency interchange timing diagram	405
Figure 62 – Recovery mechanism evaluation net	406
Figure 63 – Recovery interchange timing diagram	406
Figure 64 – Flowchart of the sub-MMS environment management state	412
Figure 65 – Domain management state chart	441
Figure 66 – Domain upload flowchart	443
Figure 67 – Domain download sequence diagram	444
Figure 68 – Domain upload sequence diagram	444
Figure 69 – Program invocation state chart	457
Figure 70 – A_Associate service procedure	495
Figure 71 – A_Release service procedure	498
Figure 72 – A_Abort service procedure	499
Figure 73 – A_Data service procedure	501
Figure 74 – A_Unidata service procedure	504
Figure 75 – Associated mode service state chart	505
Figure 76 – Non-associated mode service state chart	506
Table 1 – PERSISTDEF	36
Table 2 – VARTYPE	37
Table 3 – ITEMQUALITYDEF	38
Table 4 – STATEDEF	42
Table 5 – GROUPEXCEPTIONDEF	42
Table 6 – ACCESSRIGHTSDEF	42
Table 7 – HRESULT	43
Table 8 – UUID	50
Table 9 – Data type names for value	66
Table 10 – UUID	68
Table 11 – Create service parameters	70
Table 12 – Delete service parameters	71
Table 13 – Get attributes service parameters	72

Table 14 – Set attributes service parameters.....	74
Table 15 – Begin set attributes	75
Table 16 – End set attributes	76
Table 17 – Subscribe service parameters.....	85
Table 18 – Identify.....	88
Table 19 – Get status	89
Table 20 – Status notification.....	90
Table 21 – Initiate.....	91
Table 22 – Terminate.....	94
Table 23 – Conclude.....	96
Table 24 – Reject	96
Table 25 – Conveyance of service primitives by AREP role.....	99
Table 26 – Valid combinations of AREP roles involved in an AR	99
Table 27 – AR-Unconfirmed send.....	105
Table 28 – AR-Confirmed send	107
Table 29 – AR-Establish service	109
Table 30 – Valid combinations of AREP classes to be related	111
Table 31 – AR-Deestablish service	112
Table 32 – AR-Abort.....	113
Table 33 – AR-Compel service.....	114
Table 34 – AR-Get buffered message service.....	115
Table 35 – AR-Schedule communication service	116
Table 36 – AR-Cancel scheduled sequence service.....	117
Table 37 – AR-Status	118
Table 38 – AR-XON-OFF	119
Table 39 – AR-Remote read service.....	120
Table 40 – AR-Remote write service	121
Table 41 – Read service parameters	130
Table 42 – Read list service parameters.....	132
Table 43 – Write service parameters	135
Table 44 – Write list service parameters.....	137
Table 45 – Information report service	139
Table 46 – Information report list service.....	140
Table 47 – Exchange service parameters	143
Table 48 – Exchange list service parameters.....	146
Table 49 – Acknowledge event.....	158
Table 50 – Acknowledge event list service parameters	159
Table 51 – Enable event	161
Table 52 – Event notification service parameters.....	162
Table 53 – Enable event list.....	164
Table 54 – Notification recovery service parameters.....	165
Table 55 – Get event summary service parameters	166
Table 56 – Get event summary list service parameters	168

Table 57 – Query event summary list service parameters	171
Table 58 – Initiate load service parameters	178
Table 59 – Terminate load service parameters	180
Table 60 – Push segment service parameters	181
Table 61 – Pull segment service parameters	182
Table 62 – Discard service parameters.....	183
Table 63 – Pull upload sequencing of service primitives.....	185
Table 64 – Pull upload service parameter constraints	185
Table 65 – Pull upload state table	186
Table 66 – Pull download sequencing of service primitives	187
Table 67 – Pull download service parameter constraints	187
Table 68 – Pull download state table	188
Table 69 – Push download sequencing of service primitives	190
Table 70 – Push download service parameter constraints	190
Table 71 – Push download state table	191
Table 72 – Start service parameters	198
Table 73 – Stop service parameters	199
Table 74 – Resume service parameters.....	200
Table 75 – Reset service parameters	201
Table 76 – Kill service parameters	202
Table 77 – Action invoke service parameters.....	203
Table 78 – Action return service parameters.....	204
Table 79 – State transitions for a function invocation object	206
Table 80 – FAL class summary	233
Table 81 – Services by AREP role.....	234
Table 82 – Scope of Invoke Id.....	245
Table 83 – Types of misordering detectable by message numbers	246
Table 84 – Delivery of misordered message types on publisher/subscriber VCRs	246
Table 85 – Statistics gathered per VCR	247
Table 86 – Determination of misordering type at a subscriber VCR	247
Table 87 – Mapping of received messages to primitives.....	248
Table 88 – Mapping of received primitives to messages.....	248
Table 89 – Defined network addresses	251
Table 90 – Use of network addresses.....	251
Table 91 – Use of endpoint selectors in server VCRs.....	252
Table 92 – Use of endpoint selectors in publisher VCRs	253
Table 93 – Use of endpoint selectors in source VCRs	253
Table 94 – Network address and port numbers for device annunciation.....	254
Table 95 – Network address and port numbers for set/clear assignment info and clear address.....	255
Table 96 – Network address and port numbers for SM identify	255
Table 97 – Network address and port numbers for SM find tag.....	255
Table 98 – Network address and port numbers for clients and servers (part 1)	255

Table 99 – Network address and port numbers for clients and servers (part 2)	255
Table 100 – Network address and port numbers for publishers and subscribers	256
Table 101 – Network address and port numbers for report distribution	256
Table 102 – Network address and port numbers for LAN redundancy get and put information.....	256
Table 103 – Network address and port numbers for LAN redundancy diagnostics.....	256
Table 104 – VCR types	258
Table 105 – Use of VCR user id	258
Table 106 – Use of FDA address	259
Table 107 – Initiate	261
Table 108 – Find tag query service parameters	266
Table 109 – SMK IDs	267
Table 110 – Find tag reply service parameters	268
Table 111 – Identify service parameters	271
Table 112 – Annunciate service parameters	274
Table 113 – Set assignment info service parameters	276
Table 114 – Clear assignment info service parameters	279
Table 115 – Clear address service parameters	281
Table 116 – Diagnostic message service.....	286
Table 117 – Get redundancy info service	287
Table 118 – Put redundancy info service	289
Table 119 – Get redundancy statistics service	291
Table 120 – Open session service	298
Table 121 – Idle session service	301
Table 122 – FAL class summary	302
Table 123 – Services by AREP role.....	303
Table 124 – Access protection	319
Table 125 – Binary time coding	334
Table 126 – Asynchronous promptness events and actions	340
Table 127 – Synchronous promptness events and actions	341
Table 128 – Punctual promptness events and actions.....	343
Table 129 – Asynchronous refreshment events and actions	346
Table 130 – Synchronous refreshment events and actions.....	347
Table 131 – Punctual refreshment events and actions	349
Table 132 – A_Readloc service parameters.....	350
Table 133 – A_Writeloc service parameters.....	352
Table 134 – A_Update service parameters	354
Table 135 – A_Readfar service parameters	355
Table 136 – A_Writefar service parameters	358
Table 137 – A_Sent service parameters	359
Table 138 – A_Received service parameters.....	360
Table 139 – A_Read service parameters	362
Table 140 – A_Write service parameters	364
Table 141 – Asynchronous refreshment private mechanism events and actions	373

Table 142 – Asynchronous refreshment public mechanism events and actions	374
Table 143 – Synchronous refreshment private mechanism events and actions	375
Table 144 – Synchronous refreshment public mechanism events and actions.....	376
Table 145 – Punctual refreshment private mechanism events and actions	378
Table 146 – Punctual refreshment public mechanism events and actions	379
Table 147 – Asynchronous promptness public mechanism events and actions	382
Table 148 – Asynchronous promptness private mechanism events and actions	383
Table 149 – Synchronous promptness public mechanism events and actions	384
Table 150 – Synchronous promptness privatemechanism events and actions.....	386
Table 151 – Punctual promptness public mechanism events and actions.....	387
Table 152 – Punctual promptness privatemechanism events and actions	388
Table 153 – A_Readlist service parameters.....	397
Table 154 – Confirmed initiate service parameters	416
Table 155 – Detailed structure of the extension calling parameter.....	417
Table 156 – Detailed structure of the init request detail parameter	418
Table 157 – Detailed structure of the extension called parameter.....	419
Table 158 – Detailed structure of the init request detail parameter	420
Table 159 – Conclude service parameter.....	421
Table 160 – Unconfirmed abort service parameters	423
Table 161 – Unconfirmed reject service parameters.....	424
Table 162 – Confirmed status service parameters	426
Table 163 – Unconfirmed unsolicited status service parameter	426
Table 164 – Confirmed identify service parameters.....	427
Table 165 – Confirmed get name list service paramaters	428
Table 166 – Access group attribute description for domain object	430
Table 167 – Access rights attribute description for domain object.....	431
Table 168 – Confirmed delete domain service parameters	431
Table 169 – Confirmed initate download sequence service parameters	432
Table 170 – Confirmed download segment service parameters	434
Table 171 – Confirmed terminate download sequence service parameters	435
Table 172 – Confirmed initiate upload sequence service parameters.....	436
Table 173 – Confirmed upload segment service parameters	437
Table 174 – Confirmed terminate upload sequence service parameters	438
Table 175 – Confirmed get domain attributes service parameters.....	439
Table 176 – Access group attribute details for program invocation object.....	446
Table 177 – Access rights attribute details for program invocation object	447
Table 178 – Confirmed create program invocation service parameters	448
Table 179 – Confirmed delete program invocation service parameters	449
Table 180 – Confirmed start service parameters.....	450
Table 181 – Confirmed stop service parameters	451
Table 182 – Confirmed resume service parameters	452
Table 183 – Confirmed reset service parameters	453
Table 184 – Confirmed kill service parameters	454

Table 185 – Access group attribute details for variable object.....	459
Table 186 – Access rights attribute details for variable object	460
Table 187 – Access group attribute details for variable list object	461
Table 188 – Access right attribute details for variable list objects.....	461
Table 189 – Confirmed read service parameters.....	462
Table 190 – Confirmed write service parameters	464
Table 191 – Unconfirmed information report service parameters	465
Table 192 – Confirmed define variable-list service parameters.....	466
Table 193 – Confirmed delete variable-list service parameters.....	468
Table 194 – Confirmed get variable access attributes service parameters	469
Table 195 – Confirmed get variable-list attributes service parameters	470
Table 196 – Data type specification	472
Table 197 – Variable access specification	473
Table 198 – Variable access description attribute details	473
Table 199 – Path selection parameters	474
Table 200 – Access group attribute detail for event object	477
Table 201 – Access rights attribute details for event object.....	477
Table 202 – Unconfirmed event notification service parameters	478
Table 203 – Event type parameter details	479
Table 204 – Confirmed acknowledged event notification service parameter.....	480
Table 205 – Confirmed alter event condition monitoring service parameters.....	482
Table 206 – Confirmed get alarm summary service parameters	483
Table 207 – Confirmed get event condition attributes service parameters	485
Table 208 – Classification of service quality parameters	488
Table 209 – Identification parameters	492
Table 210 – List of MCS AR ASE services.....	493
Table 211 – A_Associate service parameters	493
Table 212 – A_Release service parameters.....	498
Table 213 – A_Abort service parameters.....	499
Table 214 – A_Data service parameters.....	500
Table 215 – A_Unidata service parameters	501

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 5-5: Application layer service definition – Type 5 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 5 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 5 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 2 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.

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 does it constrain the implementations of application layer entities within industrial automation systems.

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill the Type 5 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/TR 61158-1 (Ed.2.0), *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

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

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model – Part 3: Naming and addressing*

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

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

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

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

3 Terms and definitions

For the purposes of this document, the following terms as defined in these publications apply:

3.1 ISO/IEC 7498-1 terms

- 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

3.2 ISO/IEC 8822 terms

- a) abstract syntax
- b) presentation context

3.3 ISO/IEC 9545 terms

- 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

3.4 ISO/IEC 8824 terms

- a) object identifier
- b) type

3.5 Fieldbus data-link layer terms

For the purposes of this document, the following terms as defined in IEC 61158-3-1 and IEC 61158-4-1 apply.

- a) DL-Time
- b) DL-Scheduling-policy
- c) DLCEP
- d) DLC
- e) DL-connection-oriented mode
- f) DLPDU
- g) DLSDU
- h) DLSAP
- i) fixed tag

- j) generic tag
- k) link
- l) MAC ID
- m) network address
- n) node address
- o) node
- p) tag
- q) scheduled
- r) unscheduled

3.6 Fieldbus application layer specific definitions

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

3.6.1

access protection

limitation of the usage of an application object to one client

3.6.2

active connection control object

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

3.6.3

address assignment table

mapping of the client's internal I/O-Data object storage to the decentralised input and output data objects

3.6.4

allocate

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

3.6.5

application

function or data structure for which data is consumed or produced

3.6.6

application layer interoperability

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

3.6.7

application objects

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

3.6.8

application process

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

3.6.9

application process identifier

distinguishes multiple application processes used in a device

3.6.10

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.

3.6.11

application process object class

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

3.6.12

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

3.6.13

application relationship application service element

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

3.6.14

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.

3.6.15

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 behaviour of an object. Attributes are divided into class attributes and instance attributes.

3.6.16

behaviour

indication of how an object responds to particular events

3.6.17

bit-no

designates the number of a bit in a bitstring or an octet

3.6.18

channel

single physical or logical link of an input or output application object of a server to the process

3.6.19

channel related diagnosis

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

EXAMPLE: validity of data

3.6.20

class

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

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

3.6.21

class attributes

attribute that is shared by all objects within the same class

3.6.22

class code

unique identifier assigned to each object class

3.6.23

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.

3.6.24

client

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

3.6.25

configuration check

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

3.6.26

configuration data base

interconnection information maintained by the ACCO ASE

3.6.27

configuration fault

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

3.6.28

configuration identifier

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

3.6.29

connection

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

NOTE Connections may be either point-to-point or multipoint.

3.6.30

connection channel

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

3.6.31

connection ID (CID)

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

3.6.32

connection path

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

3.6.33

connection point

buffer which is represented as a subinstance of an Assembly object

3.6.34

consume

act of receiving data from a producer

3.6.35

consumer

node or sink that is receiving data from a producer

3.6.36

consuming application

application that consumes data

3.6.37

control commands

action invocations transferred from client to server to clear outputs, freeze inputs and/or synchronise outputs

3.6.38

conveyance path

unidirectional flow of APDUs across an application relationship

3.6.39

cyclic

repetitive in a regular manner

3.6.40

data consistency

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

3.6.41

dedicated AR

AR used directly by the FAL User

NOTE On Dedicated ARs, only the FAL Header and the user data are transferred.

3.6.42

device

physical hardware connected to the link

NOTE A device may contain more than one node.

3.6.43

device profile

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

3.6.44

diagnosis information

all data available at the server for maintenance purposes

3.6.45

diagnosis information collection

system diagnosis information that is assembled at the client side

3.6.46

dynamic AR

AR that requires the use of the AR establishment procedures to place it into an established state

3.6.47

end node

producing or consuming node

3.6.48

endpoint

one of the communicating entities involved in a connection

3.6.49

engineering

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

3.6.50

error

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

3.6.51

error class

general grouping for related error definitions and corresponding error codes

3.6.52

error code

identification of a specific type of error within an error class

3.6.53

event

an instance of a change of conditions

3.6.54

FAL subnet

subnetworks composed of one or more data link segments, identified by a subset of the network address

NOTE FAL subnets are permitted to contain bridges but not routers.

3.6.55

FIFO variable

a Variable Object class, composed of a set of homogeneously typed elements, where the first written element is the first element that can be read

NOTE On the fieldbus only one, complete element can be transferred as a result of one service invocation.

3.6.56

frame

denigrated synonym for DLPDU

3.6.57

freeze

function at the DP-slaves for simultaneous data transfer between the input data object and the process

3.6.58

group

a) <general> a general term for a collection of objects.

Specific uses:

b) <addressing> when describing an address, an address that identifies more than one entity

3.6.59

interface

a) shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

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

3.6.60

interface definition language

syntax and semantics of describing service parameters in a formal way

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

3.6.61

interface pointer

key attribute that unambiguously addresses an object interface instance

3.6.62

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.

3.6.63

I/O data

object designated to be transferred cyclically for the purpose of processing

3.6.64

identifier related diagnosis

information dedicated to modules for maintenance purpose

3.6.65

index

address of an object within an application process

3.6.66

instance

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

EXAMPLE California is an instance of the object class US-state.

NOTE The terms object, instance, and object instance are used to refer to a specific instance.

3.6.67

instance attributes

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

3.6.68

instantiated

object that has been created in a device

3.6.69

logical device

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

3.6.70

manufacturer ID

identification of each product manufacturer by a unique number

3.6.71

management information

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

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

3.6.72

member

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

3.6.73

message router

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

3.6.74

method

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

3.6.75

module

a) <general> hardware or logical component of a physical device

b) <Type 3> addressable unit inside the DP-slave

3.6.76

multipoint connection

connection from one node to many

NOTE Multipoint connections allow messages from a single producer to be received by many consumer nodes.

3.6.77

network

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

3.6.78

object

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

3.6.79

object remote procedure call

model for object oriented or component based remote method invocation

3.6.80

object specific service

service unique to the object class which defines it

3.6.81

originator

client responsible for establishing a connection path to the target

3.6.82

peer

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

3.6.83

physical device

automation or other network device

3.6.84

point-to-point connection

connection that exists between exactly two application objects

3.6.85

pre-defined AR endpoint

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

NOTE Pre-defined ARs that are not pre-established are established before being used

3.6.86

pre-established AR endpoint

AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.6.87

process data

object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing

3.6.88

produce

act of sending data to be received by a consumer

3.6.89

producer

node that is responsible for sending data

3.6.90

property

general term for descriptive information about an object

3.6.91

provider

source of a data connection

3.6.92

providerID

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

3.6.93

publisher

role of an AR endpoint that transmits APDUs onto the fieldbus for consumption by one or more subscribers

NOTE A publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated AR.

3.6.94

publishing manager

role of an AR endpoint in which it issues one or more confirmed service request APDUs to a publisher to request the publisher to publish a specified object. Two types of publishing managers are defined by this standard, pull publishing managers and push publishing managers, each of which is defined separately

3.6.95

pull publisher

type of publisher that publishes an object in response to a request received from its pull publishing manager

3.6.96

pull publishing manager

type of publishing manager that requests that a specified object be published in a corresponding response APDU

3.6.97

push publisher

type of publisher that publishes an object in an unconfirmed service request APDU

3.6.98

push publishing manager

type of publishing manager that requests that a specified object be published using an unconfirmed service

3.6.99

pull subscriber

type of subscriber that recognizes received confirmed service response APDUs as published object data

3.6.100

push subscriber

type of subscriber that recognizes received unconfirmed service request APDUs as published object data

3.6.101

quality code

additional status information of a data item

3.6.102

resource

a processing or information capability of a subsystem

3.6.103

route endpoint

object container containing Variable Objects of a variable class

3.6.104

runtime object model

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

3.6.105

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

3.6.106

service

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

3.6.107

slot

address of a module within a DP-slave

3.6.108

subscriber

role of an AREP in which it receives APDUs produced by a publisher

3.6.109

sync

function at the DP-slaves for simultaneous data transfer between the output data object and the process

3.6.110

target

end-node to which a connection is established

3.6.111

unconnected service

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

3.7 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
DLCEP	Data-link Connection End Point
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)
O⇒T	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)
T⇒O	Target to originator (connection characteristics)
VAO	Variable Object

3.8 Conventions

3.8.1 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.

3.8.2 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:

FAL ASE:	ASE Name
CLASS:	Class Name
CLASS ID:	#
PARENT CLASS:	Parent Class Name
ATTRIBUTES:	
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)
SERVICES:	
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 an 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.

3.8.3 Conventions for service definitions

3.8.3.1 General

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

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

3.8.3.2 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.

NOTE 1 See the note under 3.8.3.3 relative to the non-inclusion of service parameters that are appropriate to a protocol specification or programming interface specification or implementation specification, but not to an abstract service definition.

This standard uses a tabular format to describe the component parameters of the service primitives. The parameters that apply to each group of service primitives are set out in tables throughout the remainder of this standard. Each table consists of up to six columns: a column for the name of the service parameter, and a column each for those primitives and parameter-transfer directions used by the service. The possible six columns are

- 1) the parameter name;
- 2) the request primitive's input parameters;
- 3) the request primitive's output parameters;

NOTE 2 This is a seldom-used capability. Unless otherwise specified, request primitive parameters are input parameters.

- 4) the indication primitive's output parameters;
- 5) the response primitive's input parameters; and
- 6) the confirm primitive's output parameters.

NOTE 3 The request, indication, response and confirm primitives are also known as requestor.submit, acceptor.deliver, acceptor.submit, and requestor.deliver primitives, respectively (see ISO/IEC 10731).

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.

3.8.3.3 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.

NOTE The IEC 61158-5 series of standards define sets of abstract services. They are neither protocol specifications nor implementation specifications nor concrete programming interface specifications. Therefore there are restrictions on the extent to which service procedures can be mandated in the parts of IEC 61158-5. Protocol aspects that can vary among

different protocol specifications or different implementations that instantiate the same abstract services are unsuitable for inclusion in these service definitions, except at the level of abstraction that is necessarily common to all such expressions.

For example, the means by which service providers pair request and reply PDUs is appropriate for specification in an IEC 61158-6 protocol specification standard but not in an IEC 61158-5 abstract service definition standard. Similarly, local implementation methods by which a service provider or service user pairs request and confirm(ation) primitives, or indication and response primitives, is appropriate for an implementation specification or for a programming interface specification, but not for an abstract service standard or for a protocol standard, except at a level of abstraction that is necessarily common to all embodiments of the specifying standard. In all cases, the abstract definition is not permitted to over-specify the more concrete instantiating realization.

Further information on the conceptual service procedures of an implementation of a protocol that realizes the services of one of the IEC 61158-5 abstract service definitions can be found in IEC/TR 61158-1 (Ed.2.0), 9.6.

4 Concepts

The common concepts and templates used to describe the application layer service in this standard are detailed in IEC/TR 61158-1, Clause 9.

5 Data type ASE

5.1 Overview

An overview of the data type ASE and the relationships between data types is provided in IEC/TR 61158-1, 10.1.

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.

FAL ASE:		DATA TYPE ASE
CLASS:	DATA TYPE	
CLASS ID:		5 (FIXED LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)
PARENT CLASS:		TOP
ATTRIBUTES:		
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.

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 BOOL

This IEC 61131-3 type is the same as Boolean.

5.3.1.1.3 VT_BOOLEAN

CLASS:	Data type
ATTRIBUTES:	
2 Data type Name	= VT_BOOLEAN
4 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 Interger16).

5.3.1.2 Bitstring types

5.3.1.2.1 BitString8

CLASS:	Data type
ATTRIBUTES:	
1 Data type Numeric Identifier	= 22
2 Data type Name	= Bitstring8
3 Format	= FIXED LENGTH
5.1 Octet Length	= 1

This type contains 1 element of type BitString.

5.3.1.2.2 OCTET

This IEC 61131-3 type is the same as Bitstring8.

5.3.1.2.3 BitString16

CLASS:	Data type
ATTRIBUTES:	
1 Data type Numeric Identifier	= 23
2 Data type Name	= Bitstring16
3 Format	= FIXED LENGTH

third element is an Unsigned8 data type and gives the fraction of a day in hours. 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.

5.3.1.4.2 BinaryDate2000

CLASS:		Data type
ATTRIBUTES:		
1	Data type Numeric Identifier	= 51
2	Data type Name	= BinaryDate2000
3	Format	= FIXED LENGTH
4.1	Octet Length	= 8

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. 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 Unsigned16 data type and gives the year.

5.3.1.4.3 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.

5.3.1.4.4 DATE

CLASS:		Data type
ATTRIBUTES:		
1	Data type Numeric Identifier	= not used
2	Data type Name	= DATE
3	Format	= FIXED LENGTH
4.1	Octet Length	= 2

This IEC 61131-3 type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of two octets. It expresses the date as a number of days, starting from 1972.01.01 (January 1st, 1972), the start of the Coordinated Universal Time (UTC) era, until 2151.06.06 (June 6th, 2151), i.e. a total range of 65536 days.

5.3.1.4.5 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 1 January 0100 and 31 December 9999. The value 0,0 has been defined for 30 December 1899, 00:00. The integer part of the value represents the days after 30 December 1899 (for dates before this day, the corresponding value is negative); the fractional part defines the time at that day.

5.3.1.4.6 TimeOfDay

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
4	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.

5.3.1.4.7 TimeOfDay with date indication

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

5.3.1.4.8 TimeOfDay without date indication

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	52
2	Data type Name	=	TimeOfDay without date indication
4	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. The element is an Unsigned32 data type and gives the time after the midnight in milliseconds.

5.3.1.4.9 TIME_OF_DAY

This IEC 61131-3 type is the same as TimeofDay without date indication.

5.3.1.4.10 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.

5.3.1.4.11 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 days.

5.3.1.4.12 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.

5.3.1.4.13 TimeValue

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 21
2	Data type Name	= Time Value
3	Format	= FIXED LENGTH
4.1	Octet Length	= 8

This simple type expresses the time or time difference in a two's complement binary number with a length of eight octets. The unit of time is 1/32 millisecond.

5.3.1.4.14 UniversalTime

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 16
2	Data type Name	= UniversalTime
3	Format	= FIXED LENGTH
4.1	Octet Length	= 12

This simple type is composed of twelve elements of type VisibleString. (YYMMDDHHMMSS). It is the same as that defined in ISO/IEC 8824, except that the local time differential is not supported.

5.3.1.4.15 FieldbusTime

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 17
2	Data type Name	= FieldbusTime
3	Format	= FIXED LENGTH
4.1	Octet Length	= 7

This data type is defined in IEC 61158-4 of this standard as DL-Time.

5.3.1.5 Enumerated types

5.3.1.5.1 PERSISTDEF

CLASS: Data type

ATTRIBUTES:

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:

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	VT_BOOLEAN
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_CY	Currency
VT_DATE	date
VT_BSTR	address of a BSTR
VT_SAFEARRAY_BOOL	address of a SAFEARRAY (VT_BOOLEAN)
VT_SAFEARRAY_I1	address of a SAFEARRAY (char)
VT_SAFEARRAY_I2	address of a SAFEARRAY (short)
VT_SAFEARRAY_I4	address of a SAFEARRAY (long)
VT_SAFEARRAY_UI1	address of a SAFEARRAY (unsigned char) specified.
VT_SAFEARRAY_UI2	address of a SAFEARRAY (unsigned short) specified.
VT_SAFEARRAY_UI4	address of a SAFEARRAY (unsigned long) specified.
VT_SAFEARRAY_R4	address of a SAFEARRAY (float)
VT_SAFEARRAY_R8	address of a SAFEARRAY (double)
VT_SAFEARRAY_CY	address of a SAFEARRAY (Currency)
VT_SAFEARRAY_DATE	address of a SAFEARRAY (date)
VT_SAFEARRAY_BSTR	address of a SAFEARRAY (BSTR)
VT_DISPATCH	Interface Pointer to an IDispatch interface
VT_UNKNOWN	Interface Pointer to an IUnknown interface
VT_USERDEFINED	address of an userdefined struct
VT_ERROR	A HRESULT is specified.
NOTE All defined structured data types should have the type code VT_USERDEFINED.	

5.3.1.5.3 ITEMQUALITYDEF

CLASS: Data type

ATTRIBUTES:

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 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
	BadCommFailure	Set if there has never been any communication with this value since it was last Out of Service.
	BadCommFailureLL	and low limited
	BadCommFailureHL	and high limited
	BadCommFailureC	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.

Quality	Value	Description
	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.
	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

Quality	Value	Description
	UncertainSensorCalibration	Set during the active calibration process together with the current measured value.
	UncertainSensorCalibrationLL	and low limited
	UncertainSensorCalibrationHL	and high limited
	UncertainSensorCalibrationC	and constant
Good	GoodNonCascOk	No error or special condition is associated with this value.
Non Cascade	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 Alarm with a priority greater than or equal to 8.
	GoodNonCascUnackCriticalAlarmLL	and low limited
	GoodNonCascUnackCriticalAlarmHL	and high limited
	GoodNonCascUnackCriticalAlarmC	and constant
	GoodNonCascInitialFailSafe	This value is from a block that wants its following output block (e.g. AO) to go to Fail Safe.
	GoodNonCascInitialFailSafeLL	and low limited
	GoodNonCascInitialFailSafeHL	and high limited
	GoodNonCascInitialFailSafeC	and constant
	GoodNonCascMaintenanceRequired	The device works still without failure but service support will be necessary soon. This may be detected e.g. by a Transducer Block of a value of pH meter.
	GoodNonCascMaintenanceRequiredLL	and low limited
	GoodNonCascMaintenanceRequiredHL	and high limited

Quality	Value	Description
	GoodNonCascMaintenanceRequiredC	and constant
Good	GoodCascOk	No error or special condition is associated with this value.
Cascade	GoodCascOkC	and constant
	GoodCascInitializationAcknowledge	The value is an initialized value from a source (cascade input, remote-cascade in, and remote-output in parameters).
	GoodCascInitializationAcknowledgeLL	and low limited
	GoodCascInitializationAcknowledgeHL	and high limited
	GoodCascInitializationAcknowledgeC	and constant
	GoodCascInitializationRequest	The value is an initialization value for a source (back calculation input parameter), because the lower loop is broken or the mode is wrong.
	GoodCascInitializationRequestLL	and low limited
	GoodCascInitializationRequestHL	and high limited
	GoodCascInitializationRequestC	and constant
	GoodCascNotInvited	The value is from a block which does not have a target mode that would use this input. This covers all cases other than Fail Safe Active, Local Override, and Not Selected. The target mode can be the next permitted mode of higher priority in the case of shedding a supervisory computer.
	GoodCascNotInvitedLL	and low limited
	GoodCascNotInvitedHL	and high limited
	GoodCascNotInvitedC	and constant
	GoodCascDoNotSelect	The value is from a block which should not be selected, due to conditions in or above the block.
	GoodCascDoNotSelectLL	and low limited
	GoodCascDoNotSelectHL	and high limited
	GoodCascDoNotSelectC	and constant
	GoodCascLocalOverride	The value is from a block that has been locked out by a local key switch or is a Complex AO/DO with interlock logic active. The failure of normal control should be propagated to a PID block for alarm and display purposes. This also implies Not Invited.
	GoodCascLocalOverrideLL	and low limited
	GoodCascLocalOverrideHL	and high limited
	GoodCascLocalOverrideC	and constant
	GoodCascInitiateFailSafe	The value is from a block that wants its downstream output block (e.g. AO) to go to Fail Safe. This is determined by a block option to initiate Fail Safe if the status of the primary input and/or cascade input goes Bad.
	GoodCascInitiateFailSafeLL	and low limited
	GoodCascInitiateFailSafeHL	and high limited
	GoodCascInitiateFailSafeC	and constant

5.3.1.5.4 STATEDEF

CLASS: Data type

ATTRIBUTES:

2 Data type Name = STATEDEF
3 Format = FIXED LENGTH
4.1 Octet Length = 2

The allowed values are shown in Table 4.

Table 4 – STATEDEF

Value
CBANonExistent
CBAInitializing
CBAReady
CBAOperating
CBADefect

5.3.1.5.5 GROUPEXCEPTIONDEF

CLASS: Data type

ATTRIBUTES:

2 Data type Name = GROUPEXCEPTIONDEF

3 Format = FIXED LENGTH

4.1 Octet Length = 2

The allowed values are shown in Table 5.

Table 5 – GROUPEXCEPTIONDEF

Value
CBANonAccessible
CBAOkay
CBAProblem
CBAUnknown

5.3.1.5.6 ACCESSRIGHTSDEF

CLASS: Data type

ATTRIBUTES:

2 Data type Name = ACCESSRIGHTSDEF

3 Format = FIXED LENGTH

4.1 Octet Length = 2

The allowed values are shown in Table 6.

Table 6 – ACCESSRIGHTSDEF

Value
CBANoAccess
CBAReadAccess
CBAWriteAccess
CBAFullAccess

5.3.1.6 Handle types

5.3.1.6.1 HRESULT

CLASS: Data type

ATTRIBUTES:

2 Data type Name = HRESULT

3 Format = FIXED LENGTH

4.1 Octet Length = 4

The allowed values are shown in Table 7. The defined HRESULT values may be extended by user specific values. If necessary the coding scheme as defined in IEC 61158-6-5 should be applied. In particular, the severity code to indicate success or error should be used.

Table 7 – HRESULT

Value	Description
CBA_E_MALFORMED	The identifier is malformed (too long, unallowed characters, no separation characters, syntactically invalid)
CBA_E_UNKNOWNOBJECT	The object specified in the item identifier is not known.
CBA_E_UNKNOWNMEMBER	The member specified in the item identifier is not known.
CBA_E_TYPERISMATCH	The type specified does not match the expected type.
CBA_E_INVALIDENUMVALUE	The enumeration value is invalid.
CBA_E_INVALIDID	The ConsumerID or ProviderID is not valid.
CBA_E_INVALIDEPSILON	The Epsilon type or value is not valid.
CBA_E_INVALIDSUBSTITUTE	The Substitute type or value is not valid.
CBA_E_INVALIDCONNECTION	It is not allowed to specify a connection from an identifier to itself.
CBA_E_INVALIDCOOKIE	The Cookie value is not valid.
CBA_E_TIMEVALUEUNSUPPORTED	The time value is unsupported.
CBA_E_QOSTYPEUNSUPPORTED	The QoS type is unsupported.
CBA_E_QOSVALUEUNSUPPORTED	The QoS value is unsupported.
CBA_E_PERSISTRUNNING	While the Save service is running, no changes are allowed to the configuration data base (try again in a few seconds).
CBA_E_INUSE	The destination specified in the item identifier is already connected to some other provider.
CBA_E_NOTAPPLICABLE	The operation is currently not applicable.
CBA_E_NONACCESSIBLE	The item is not accessible.
CBA_E_DEFECT	Hardware defect detected, replacement needed.
CBA_S_PERSISTPENDING	The value requested is currently not persistent.
CBA_S_ESTABLISHING	The connection is not yet established.
CBA_S_NOCONNECTION	There is no connection from the provider to this specific consumer.
CBA_S_VALUEBUFFERED	The value was only buffered and has no immediate effect on the process (e.g. device is in state CBAReady).
CBA_S_VALUEUNCERTAIN	The value requested is currently uncertain.
E_OUTOFMEMORY	Insufficient memory to complete the call.
E_INVALIDARG	One or more arguments are invalid.
E_NOTIMPL	The service is not implemented.
E_FAIL	Unspecified error.
E_NOINTERFACE	No such interface supported.
RPC_S_PROCNUM_OUT_OF_RANGE	The procedure number is out of range.
RPC_E_INVALID_OXID	The object exporter was not found.
RPC_E_INVALID_OID	The specified object was not found or recognized.
RPC_E_INVALID_SET	The object exporter set was not found.
RPC_E_INVALID_OBJECT	The requested object does not exist.
RPC_E_VERSION_MISMATCH	The ORPC version on the client and server machine does not match.
DISP_E_BADINDEX	Invalid index.
DISP_E_BADPARAMCOUNT	The number of elements provided to DISPPARAMS is different from the number of arguments accepted by the method or property.

Value	Description
DISP_E_BADVARTYPE	One of the arguments is not a valid variant type.
DISP_E_EXCEPTION	The application needs to raise an exception. In this case, the structure passed in pExceplInfo should be filled in.
DISP_E_MEMBERNOTFOUND	The requested member does not exist, or the call to Invoke tried to set the value of a read-only property.
DISP_E_NONAMEDARGS	This implementation of IDispatch does not support named arguments
DISP_E_OVERFLOW	One of the arguments could not be coerced to the specified type.
DISP_E_PARAMNOTFOUND	One of the parameter DISPIDs does not correspond to a parameter on the method. In this case, puArgErr should be set to the first argument that contains the error.
DISP_E_TYPEMISMATCH	One or more of the arguments could not be coerced. The index within rgvarg of the first parameter with the incorrect type is returned in the puArgErr parameter.
DISP_E_UNKNOWNINTERFACE	The interface identifier passed in riid is not IID_NULL.
DISP_E_UNKNOWNLCID	Unknown language.
DISP_E_UNKNOWNNAME	Unknown name.
DISP_E_PARAMNOTOPTIONAL	A required parameter was omitted.
TYPE_E_ELEMENTNOTFOUND	Element not found.
S_OK	Success, "everything worked".
S_FALSE	Success but with additional results, "function worked and the result is false".

5.3.1.7 Numeric types

5.3.1.7.1 BCD

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	35
2	Data type Name	=	Unsigned8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This type has a length of one octet. In this type, the least significant four bits are used to express a BCD value which is between zero and nine inclusive. The most significant four bits are unused.

5.3.1.7.2 Floating Point types

5.3.1.7.2.1 Float32

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float32
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type has a length of four octets. The format for float32 is that defined by ANSI/IEEE 754 as single precision.

5.3.1.7.2.2 Floating point

This data type is the same as Float32.

5.3.1.7.2.3 REAL

This IEC 61131-3 type is the same as Float32.

5.3.1.7.2.4 float

This data type is the same as Float32.

5.3.1.7.2.5 Float64

CLASS:	Data type
ATTRIBUTES:	
1 Data type Numeric Identifier	= 15
2 Data type Name	= Float64
3 Format	= FIXED LENGTH
4.1 Octet Length	= 8

This type has a length of eight octets. The format for float64 is that defined by ANSI/IEEE 754 as double precision.

5.3.1.7.2.6 LREAL

This IEC 61131-3 type is the same as Float64.

5.3.1.7.2.7 double

This data type is the same as Float64.

5.3.1.7.3 Integer types

5.3.1.7.3.1 Integer8

CLASS:	Data type
ATTRIBUTES:	
1 Data type Numeric Identifier	= 2
2 Data type Name	= Integer8
3 Format	= FIXED LENGTH
4.1 Octet Length	= 1

This integer type is a two's complement binary number with a length of one octet.

5.3.1.7.3.2 SINT

This IEC 61131-3 type is the same as Integer8.

5.3.1.7.3.3 char

This data type is the same as Integer8.

5.3.1.7.3.4 Integer16

CLASS:	Data type
ATTRIBUTES:	
1 Data type Numeric Identifier	= 3
2 Data type Name	= Integer16
3 Format	= FIXED LENGTH

4.1 Octet Length = 2

This integer type is a two's complement binary number with a length of two octets.

5.3.1.7.3.5 INT

This IEC 61131-3 type is the same as Integer16.

5.3.1.7.3.6 short

This data type is the same as Integer16.

5.3.1.7.3.7 Integer32

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	4
2	Data type Name	=	Integer32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is a two's complement binary number with a length of four octets.

5.3.1.7.3.8 DINT

This IEC 61131-3 type is the same as Integer32.

5.3.1.7.3.9 long

This data type is the same as Integer32.

5.3.1.7.3.10 Integer64

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	55
2	Data type Name	=	Integer64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This integer type is a two's complement binary number with a length of eight octets.

5.3.1.7.3.11 LINT

This IEC 61131-3 type is the same as Integer64.

5.3.1.7.4 Unsigned types

5.3.1.7.4.1 Unsigned8

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	5
2	Data type Name	=	Unsigned8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This type has a length of one octet.