DEUTSCHE NORM **Entwurf** Juni 2009

DIN IEC 61158-600

DIN

ICS 35.100.70; 35.160; 35.240.50

Einsprüche bis 2009-08-31
Ersatzvermerk
siehe unten

Entwurf

Industrielle Kommunikationsnetze – Feldbusse – Teil 600: Protokollspezifikation des Application Layer (Anwendungsschicht); Englische Fassung (IEC 65C/526/CD:2009), nur auf CD-ROM

Industrial communication networks –
Fieldbus specifications –
Part 600: Application Layer protocol specification;
English version (IEC 65C/526/CD:2009), only on CD-ROM

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2009-06-29 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfes besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise als Datei per E-Mail an dke@din.de in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (Hausanschrift: Stresemannallee 15, 60596 Frankfurt am Main).

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevante Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Ersatzvermerk

Vorgesehen als Ersatz für DIN EN 61158-6-2:2008-09, DIN EN 61158-6-3:2008-09, DIN EN 61158-6-10:2008-09, DIN EN 61158-6-12:2008-09, DIN EN 61158-6-14:2008-09, DIN EN 61158-6-15:2008-09 und DIN EN 61158-6-19:2008-09

Gesamtumfang 1768 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Nationales Vorwort

Die englische Originalfassung des internationalen Dokuments IEC 65C/526/CD:2009 „Industrial communication networks – Fieldbus specifications – Part 600: Application Layer protocol specification“ (CD, en: Committee Draft) ist unverändert in diesen Norm-Entwurf übernommen worden. Der vorliegende Normentwurf besteht aus acht Teilen. Nach erfolgreicher Abstimmung ist beabsichtigt, diese einzeln als Normen zu veröffentlichen. Eine Übersicht über die Normen der Reihe DIN EN 61158 und DIN EN 61784 befindet sich in den Mitteilungen der DKE vom 29.09.2008, siehe www.dke.de, links oben in das Suchfeld „61158“ eingeben.

Das internationale Dokument wurde vom SC 65C „Industrial networks“ der Internationalen Elektrotechnischen Kommission (IEC) erarbeitet und den nationalen Komitees zur Stellungnahme vorgelegt.

Die IEC und das Europäische Komitee für Elektrotechnische Normung (CENELEC) haben vereinbart, dass ein auf IEC-Ebene erarbeiteter Entwurf für eine Internationale Norm zeitgleich (parallel) bei IEC und CENELEC zur Umfrage (CDV-Stadium) und Abstimmung als FDIS (en: Final Draft International Standard) bzw. Schluss-Entwurf für eine Europäische Norm gestellt wird, um eine Beschleunigung und Straffung der Normungsarbeit zu erreichen. Dokumente, die bei CENELEC als Europäische Norm angenommen und ratifiziert werden, sind unverändert als Deutsche Normen zu übernehmen.

Es ist vorgesehen, auch bei der entsprechenden zukünftigen Deutschen Norm auf die deutsche Sprachfassung zu verzichten und diese in der englischsprachigen Fassung zu veröffentlichen.

Da der Abstimmungszeitraum für einen FDIS bzw. Schluss-Entwurf prEN nur 2 Monate beträgt, und dann keine sachlichen Stellungnahmen mehr abgegeben werden können, sondern nur noch eine „JA/NEIN“-Entscheidung möglich ist, wobei eine „NEIN“-Entscheidung fundiert begründet werden muss, wird bereits der CD als DIN-Norm-Entwurf veröffentlicht, um die Stellungnahmen aus der Öffentlichkeit frühzeitig berücksichtigen zu können.

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

Da sich die Benutzer des vorliegenden Norm-Entwurfs der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der IEC 61158-600 veröffentlicht. Für die meisten der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Dieser Norm-Entwurf steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/2004 festgelegt, dass DIN-Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der Informationsverarbeitung gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Diese Bedingungen sind für die vorliegende Norm erfüllt.

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 EN 60068 als Europäische Norm 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 das Protokoll des Application Layer für die einzelnen Typen fest.

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Erklärung zu Schutzrechten

Die Internationale Elektrotechnische Kommission macht darauf aufmerksam, dass ihr gegenüber erklärt wurde, dass die Übereinstimmung mit dem vorgesehenen Norm-Inhalt die Anwendung von Patenten erforderlich machen könnte. Nähere Informationen hierzu befinden sich im Vorwort des vorliegenden Internationalen Norm-Entwurfes.

Es wird darauf aufmerksam gemacht, dass Bestandteile des vorliegenden Norm-Entwurfs Gegenstand von weiteren Patentrechten sein können. Es liegt nicht im Verantwortungsbereich der DKE, einige oder alle dieser Patentrechte zu benennen.

Der Leser des vorliegenden Norm-Entwurfs wird jedoch gebeten, ihm bekannte derartige Patentrechte der Geschäftsstelle mitzuteilen und entsprechende Informationen zur Verfügung zu stellen.

Änderungen

Der vorliegende Entwurf fasst die gegenüber DIN EN 61158-6-2:2008-09, DIN EN 61158-6-3:2008-09, DIN EN 61158-6-10:2008-09, DIN EN 61158-6-12:2008-09, DIN EN 61158-6-14:2008-09, DIN EN 61158-6-15:2008-09 und DIN EN 61158-6-19:2008-09 vorgesehenen Änderungen zusammen, außerdem fügt er die Feldbuselemente des Typs 21 (RapiNet) hinzu.

Gegenüber DIN EN 61158-6-2:2008-09 sind folgende Änderungen vorgesehen:

a) normative Verweisungen und Literaturhinweise auf den neuesten Stand gebracht

b) Abkürzungen in 3.4 auf den neuesten Stand gebracht

c) in 4.1.2.1 die Liste der PDUs für Dienstanforderungen und Antwort (Time Sync und Parameter ASE-Objekte) auf den neuesten Stand gebracht

d) Time Sync ASE-Objekt in 4.1.8.6 auf den neuesten Stand gebracht

e) neue Parameter für das ASE-Objekt in 4.1.8.7 eingeführt

f) kleinere Verbesserungen in den allgemeinen Parametern in 4.1.9.4

g) Objekte und Service Codes für Time Sync und Parameter ASEs hinzugefügt bzw. auf den neuesten Stand gebracht in 4.1.10.2

h) kleinere Verbesserungen der Fehlermeldungen in 4.1.11

i) neuen Unterabschnitt 11.7 mit neuen QoS-Spezifikationen hinzugefügt

Gegenüber DIN EN 61158-6-3:2008-09 sind folgende Änderungen vorgesehen:

a) Fehlerberichtigungen

b) textliche Überarbeitung

Gegenüber DIN EN 61158-6-10:2008-09 sind folgende Änderungen vorgesehen:

a) Fehlerberichtigungen

b) textliche Überarbeitung

c) Optimierung der Synchronisation und Optimierung der Startzeit nach Netz-Aus

Gegenüber DIN EN 61158-6-12:2008-09 sind folgende Änderungen vorgesehen:

a) Fehlerberichtigungen

b) textliche Überarbeitung

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Gegenüber DIN EN 61158-6-14:2008-09 sind folgende Änderungen vorgesehen:

a) Fehlerberichtigungen

b) textliche Überarbeitung

Gegenüber DIN EN 61158-6-15:2008-09 sind folgende Änderungen vorgesehen:

a) ausschließlich kleinere textliche Überarbeitungen

Gegenüber DIN EN 61158-6-19:2008-09 sind folgende Änderungen vorgesehen:

a) Fehlerberichtigungen

b) textliche Überarbeitung

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

INCLUDED SUBPARTS

NOTE The Contents of each subpart begins on page 3, so that the page numbering will be unchanged when the final International Standard is issued with its IEC-Central-Office-provided cover sheet.

IEC 61158-6-2, Industrial communication networks – Fieldbus specifications – Part 6-2: Application layer service definition – Type 2 elements 2003

IEC 61158-6-3, Industrial communication networks – Fieldbus specifications – Part 6-3: Application layer service definition – Type 3 elements 3003

IEC 61158-6-10, Industrial communication networks – Fieldbus specifications – Part 6-10: Application layer service definition – Type 10 elements 10003

IEC 61158-6-12, Industrial communication networks – Fieldbus specifications – Part 6-12: Application layer service definition – Type 12 elements 12003

IEC 61158-6-14, Industrial communication networks – Fieldbus specifications – Part 6-14: Application layer service definition – Type 14 elements 14003

IEC 61158-6-15, Industrial communication networks – Fieldbus specifications – Part 6-15: Application layer service definition – Type 15 elements 15003

IEC 61158-6-19, Industrial communication networks – Fieldbus specifications – Part 6-19: Application layer service definition – Type 19 elements 19003

IEC 61158-6-21, Industrial communication networks – Fieldbus specifications – Part 6-21: Application layer service definition – Type 21 elements 21003

The following subparts remain unchanged.

IEC 61158-6-1, The Type 1 Application layer specifications have been deleted for lack of market relevance.

IEC 61158-6-4, Industrial communication networks – Fieldbus specifications – Part 5-4: Application layer service definition – Type 4 elements

IEC 61158-6-5, Industrial communication networks – Fieldbus specifications – Part 6-4: Application layer service definition – Type 5 elements

IEC 61158-6-6, The Type 6 specifications have been deleted for lack of market relevance.

IEC 61158-6-7, Industrial communication networks – Fieldbus specifications – Part 6-7: Application layer service definition – Type 7 elements

IEC 61158-6-8, Industrial communication networks – Fieldbus specifications – Part 6-8: Application layer service definition – Type 8 elements

IEC 61158-6-9, Industrial communication networks – Fieldbus specifications – Part 6-9: Application layer service definition – Type 9 elements

IEC 61158-6-11, Industrial communication networks – Fieldbus specifications – Part 6-11: Application layer service definition – Type 11 elements

IEC 61158-6-13, Industrial communication networks – Fieldbus specifications – Part 6-13: Application layer service definition – Type 13 elements

IEC 61158-6-16, Industrial communication networks – Fieldbus specifications – Part 6-16: Application layer service definition – Type 16 elements

IEC 61158-6-17, Industrial communication networks – Fieldbus specifications – Part 6-17: Application layer service definition – Type 17 elements

IEC 61158-6-18, Industrial communication networks – Fieldbus specifications – Part 6-18: Application layer service definition – Type 18 elements

IEC 61158-6-20, Industrial communication networks – Fieldbus specifications – Part 6-20: Application layer service definition – Type 20 elements

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

CONTENTS

FOREWORD 2013
INTRODUCTION 2015
1 Scope 2016
1.1 General 2016
1.2 Specifications 2016
1.3 Conformance 2016
2 Normative references 2017
3 Terms, definitions, symbols, abbreviations and conventions 2018
3.1 Terms and definitions from other ISO/IEC standards 2018
3.2 Terms and definitions from IEC 61158-5-2 2019
3.3 Additional terms and definitions 2019
3.4 Abbreviations and symbols 2025
3.5 Conventions 2025
3.6 Conventions used in state machines 2030
4 Abstract syntax 2031
4.1 FAL PDU abstract syntax 2031
4.2 Data abstract syntax specification 2121
4.3 Encapsulation abstract syntax 2125
5 Transfer syntax 2139
5.1 Compact encoding 2139
5.2 Data type reporting 2147
6 Structure of FAL protocol state machines 2152
7 AP-Context state machine 2153
7.1 Overview 2153
7.2 Connection object state machine 2153
8 FAL service protocol machine (FSPM) 2162
8.1 General 2162
8.2 Primitive definitions 2162
8.3 Parameters of primitives 2166
8.4 FSPM state machines 2167
9 Application relationship protocol machines (ARPMs) 2168
9.1 General 2168
9.2 Connection-less ARPM (UCMM) 2168
9.3 Connection-oriented ARPMs (transports) 2178
10 DLL mapping protocol machine 1 (DMPM 1) 2249
10.1 General 2249
10.2 Link producer 2249
10.3 Link consumer 2249
10.4 Primitive definitions 2249
10.5 DMPM state machine 2251
10.6 Data-link Layer service selection 2253
11 DLL mapping protocol machine 2 (DMPM 2) 2253
11.1 General 2253
11.2 Mapping of UCMM PDUs 2253
11.3 Mapping of transport class 0 and class 1 PDUs 2258

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

11.4 Mapping of transport class 2 and class 3 PDU's 2260
11.5 Mapping of transport classes 4 to 6 2260
11.6 IGMP Usage 2260
11.7 Quality of Service (QoS) for CP 2/2 messages 2261
11.8 Management of an encapsulation session 2264
12 DLL mapping protocol machine 3 (DMPM 3) 2265
Bibliography 2266

Figure 1 – Attribute table format and terms 2026
Figure 2 – Service request/response parameter 2026
Figure 3 – Example of an STD 2030
Figure 4 – Network connection parameters 2050
Figure 5 – Time tick 2052
Figure 6 – Connection establishment time-out 2054
Figure 7 – Transport Class Trigger attribute 2087
Figure 8 – CP2/3_initial_comm_characteristics attribute format 2090
Figure 9 – Segment type 2098
Figure 10 – Port segment 2099
Figure 11 – Logical segment encoding 2101
Figure 12 – Extended network segment 2105
Figure 13 – Encapsulation message 2126
Figure 14 – FixedLengthBitString compact encoding bit placement rules 2143
Figure 15 – Example compact encoding of a SWORD FixedLengthBitString 2144
Figure 16 – Example compact encoding of a WORD FixedLengthBitString 2144
Figure 17 – Example compact encoding of a DWORD FixedLengthBitString 2144
Figure 18 – Example compact encoding of a LWORD FixedLengthBitString 2144
Figure 19 – Example 2 of formal encoding of a structure type specification 2149
Figure 20 – Example of abbreviated encoding of a structure type specification 2149
Figure 21 – Example 1 of formal encoding of an array type specification 2150
Figure 22 – Example 2 of formal encoding of an array type specification 2150
Figure 23 – Example 1 of abbreviated encoding of an array type specification 2151
Figure 24 – Example 2 of abbreviated encoding of an array type specification 2151
Figure 25 – I/O Connection object state transition diagram 2153
Figure 26 –Bridged Connection object state transition diagram 2157
Figure 27 – Explicit Messaging Connection object state transition diagram 2159
Figure 28 – State transition diagram of UCMM client 2171
Figure 29 – State transition diagram of high–end UCMM server 2172
Figure 30 – State transition diagram of low–end UCMM server 2175
Figure 31 – Sequence diagram for a UCMM with one outstanding message 2176
Figure 32 – Sequence diagram for a UCMM with multiple outstanding messages 2177
Figure 33 – TPDU buffer 2178
Figure 34 – Data flow diagram using a client transport class 0 and server transport class 0 2181
Figure 35 – Sequence diagram of data transfer using transport class 0 2181

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Figure 36 – Class 0 client STD 2182
Figure 37 – Class 0 server STD 2183
Figure 38 – Data flow diagram using client transport class 1 and server transport class 1 2184
Figure 39 – Sequence diagram of data transfer using client transport class 1 and server transport class 1 2185
Figure 40 – Class 1 client STD 2187
Figure 41 – Class 1 server STD 2188
Figure 42 – Data flow diagram using client transport class 2 and server transport class 2 2190
Figure 43 – Diagram of data transfer using client transport class 2 and server transport class 2 *without* returned data 2191
Figure 44 – Sequence diagram of data transfer using client transport class 2 and server transport class 2 *with* returned data 2192
Figure 45 – Class 2 client STD 2193
Figure 46 – Class 2 server STD 2195
Figure 47 – Data flow diagram using client transport class 3 and server transport class 3 2198
Figure 48 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 *without* returned data 2199
Figure 49 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 *with* returned data 2200
Figure 50 – Class 3 client STD 2202
Figure 51 – Class 3 server STD 2205
Figure 52 – Data flow diagram using transport classes 4 and 5 2207
Figure 53 – Sequence diagram of message exchange using transport classes 4 and 5 2208
Figure 54 – Sequence diagram of messages overwriting each other 2209
Figure 55 – Sequence diagram of queued message exchange using transport classes 4 and 5 2210
Figure 56 – Sequence diagram of retries using transport classes 4 and 5 2210
Figure 57 – Sequence diagram of idle traffic using transport classes 4 and 5 2211
Figure 58 – Classes 4 and 5 basic structure 2212
Figure 59 – Class 6 basic structure 2213
Figure 60 – Classes 4 to 6 general STD 2214
Figure 61 – Class 4 sender STD 2216
Figure 62 – Class 4 receiver STD 2219
Figure 63 – Sequence diagram of three fragments using transport class 5 2222
Figure 64 – Sequence diagram of fragmentation with retries using transport class 5 2223
Figure 65 – Sequence diagram of two fragments using transport class 5 2223
Figure 66 – Sequence diagram of aborted message using transport class 5 2224
Figure 67 – Class 5 sender STD 2225
Figure 68 – Class 5 receiver STD 2228
Figure 69 – Data flow diagram for transport class 6 2233
Figure 70 – Sequence diagram of message exchange using transport class 6 2234
Figure 71 – Sequence diagram of retries using transport class 6 2235
Figure 72 – Sequence diagram of idle traffic using transport class 6 2236

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Figure 73 – Sequence diagram of request overwriting null 2237
Figure 74 – Sequence diagram of response overwriting ACK of null 2237
Figure 75 – Sequence diagram of three fragments using transport class 6 2238
Figure 76 – Sequence diagram of fragmentation with retries using transport class 6 2239
Figure 77 – Sequence diagram of two fragments using transport class 6 2239
Figure 78 – Sequence diagram of aborted fragmented sequence using transport class 6 2240
Figure 79 – Class 6 client STD 2241
Figure 80 – Class 6 server STD 2244
Figure 81 – Data flow diagram for a link producer and consumer 2249
Figure 82 – State transition diagram for a link producer 2252
Figure 83 – State transition diagram for a link consumer 2252
Figure 84 – DS field in the IP header 2262
Figure 85 – IEEE 802.1Q tagged frame 2263

Table 1 – Get_Attribute_All response service rules 2027
Table 2 – Example class level object/service specific response data of Get_Attribute_All 2027
Table 3 – Example Get_Attribute_All data array method 2028
Table 4 – Set_Attribute_All request service rules 2029
Table 5 – Example Set_Attribute_All attribute ordering method 2029
Table 6 – Example Set_Attribute_All data array method 2029
Table 7 – State event matrix format 2031
Table 8 – Example state event matrix 2031
Table 9 – UCMM_PDU header format 2034
Table 10 – UCMM command codes 2034
Table 11 – Transport class 0 header 2035
Table 12 – Transport class 1 header 2036
Table 13 – Transport class 2 header 2036
Table 14 – Transport class 3 header 2036
Table 15 – Classes 4 to 6 header format 2036
Table 16 – Real-time data header – exclusive owner 2037
Table 17 – Real-time data header– redundant owner 2038
Table 18 – Forward_Open request format 2040
Table 19 – Forward_Open_Good response format 2041
Table 20 – Forward_Open_Bad response format 2041
Table 21 – Large_Forward_Open request format 2042
Table 22 – Large_Forward_Open_Good response format 2043
Table 23 – Large_Forward_Open_Bad response format 2043
Table 24 – Forward_Close request format 2044
Table 25 – Forward_Close_Good response format 2044
Table 26 – Forward_Close_Bad response format 2045
Table 27 – Unconnected_Send request format 2046
Table 28 – Unconnected_Send_Good response format 2046

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Table 29 – Unconnected_Send_Bad response format 2047
Table 30 – Get_Connection_Data request format 2047
Table 31 – Get_Connection_Data response format 2048
Table 32 – Search_Connection_Data request format 2048
Table 33 – Get_Object_Owner request format 2049
Table 34 – Forward_Open_Good response format 2049
Table 35 – Time-out multiplier 2052
Table 36 – Time tick units 2052
Table 37 – Selection of connection ID 2056
Table 38 – Transport class, trigger and Is_Server format 2056
Table 39 – MR_Request_Header format 2057
Table 40 – MR_Response_Header format 2057
Table 41 – Structure of Get_Attribute_All_ResponsePDU body 2058
Table 42 – Structure of Set_Attribute_All_RequestPDU body 2058
Table 43 – Structure of Get_Attribute_List_RequestPDU body 2058
Table 44 – Structure of Get_Attribute_List_ResponsePDU body 2058
Table 45 – Structure of Set_Attribute_List_RequestPDU body 2058
Table 46 – Structure of Set_Attribute_List_ResponsePDU body 2059
Table 47 – Structure of Reset_RequestPDU body 2059
Table 48 – Structure of Reset_ResponsePDU body 2059
Table 49 – Structure of Start_RequestPDU body 2059
Table 50 – Structure of Start_ResponsePDU body 2059
Table 51 – Structure of Stop_RequestPDU body 2060
Table 52 – Structure of Stop_ResponsePDU body 2060
Table 53 – Structure of Create_RequestPDU body 2060
Table 54 – Structure of Create_ResponsePDU body 2060
Table 55 – Structure of Delete_RequestPDU body 2060
Table 56 – Structure of Delete_ResponsePDU body 2060
Table 57 – Structure of Get_Attribute_Single_ResponsePDU body 2061
Table 58 – Structure of Set_Attribute_Single_RequestPDU body 2061
Table 59 – Structure of Set_Attribute_Single_ResponsePDU body 2061
Table 60 – Structure of Find_Next_Object_Instance_RequestPDU body 2061
Table 61 – Structure of Find_Next_Object_Instance_ResponsePDU body 2061
Table 62 – Structure of Apply_Attributes_RequestPDU body 2062
Table 63 – Structure of Apply_Attributes_ResponsePDU body 2062
Table 64 – Structure of Save_RequestPDU body 2062
Table 65 – Structure of Save_ResponsePDU body 2062
Table 66 – Structure of Restore_RequestPDU body 2062
Table 67 – Structure of Restore_ResponsePDU body 2062
Table 68 – Structure of Group_Sync_RequestPDU body 2063
Table 69 – Structure of Group_Sync_ResponsePDU body 2063
Table 70 – Identity object class attributes 2063
Table 71 – Identity object instance attributes 2063

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Table 72 – Identity object bit definitions for status instance attribute 2064
Table 73 – Bits 4 – 7 of status instance attribute 2065
Table 74 – Class level object/service specific response data of Get_Attribute_All 2065
Table 75 – Instance level object/service specific response data of Get_Attribute_All 2066
Table 76 – Modified instance level object/service specific response data of Get_Attribute_All 2066
Table 77 – Object-specific parameter for Reset 2067
Table 78 – Message Router object class attributes 2067
Table 79 – Message Router object instance attributes 2067
Table 80 – Class level object/service specific response data of Get_Attribute_All 2068
Table 81 – Instance level object/service specific response data of Get_Attribute_All 2068
Table 82 – Assembly object class attributes 2068
Table 83 – Assembly object instance attributes 2068
Table 84 – Acknowledge Handler object class attributes 2069
Table 85 – Acknowledge Handler object instance attributes 2070
Table 86 – Structure of Add_AckData_Path_RequestPDU body 2070
Table 87 – Structure of Remove_AckData_Path_RequestPDU body 2070
Table 88 – Time Sync object instance attributes 2071
Table 89 – ClockIdentity encoding for different network implementations 2074
Table 90 – ClockClass values 2074
Table 91 – TimeAcuracy values 2075
Table 92 – TimePropertyFlags bit values 2075
Table 93 – TimeSource values 2075
Table 94 – Types of Clock 2076
Table 95 – Parameter object class attributes 2076
Table 96 – Parameter Class Descriptor bit values 2077
Table 97 – Parameter object instance attributes 2078
Table 98 – Semantics of Descriptor Instance attribute 2078
Table 99 – Minimum and Maximum Value semantics 2080
Table 100 – Scaling Formula attributes 2081
Table 101 – Scaling links 2081
Table 102 – Structure of Get_Enum_String_RequestPDU body 2082
Table 103 – Structure of Get_Enum_String_ResponsePDU body 2082
Table 104 – Enumerated strings Type versus Parameter data type 2082
Table 105 – Connection Manager object class attributes 2083
Table 106 – Connection Manager object instance attributes 2083
Table 107 – Connection object class attributes 2084
Table 108 – Connection object instance attributes 2085
Table 109 – Values assigned to the state attribute 2086
Table 110 – Values assigned to the instance_type attribute 2086
Table 111 – Possible values within Direction Bit 2087
Table 112 – Possible values within Production Trigger Bits 2087
Table 113 – Possible values within Transport Class Bits 2088

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Table 114 – Transport Class_Trigger attribute 2089
Table 115 – Values defined for the CP2/3_produced_connection_id attribute 2089
Table 116 – Values defined for the CP2/3_consumed_connection_id attribute 2090
Table 117 – Values for the Initial Production Characteristics nibble 2091
Table 118 – Values for the Initial Consumption Characteristics nibble 2092
Table 119 – Values for the watchdog_timeout_action 2095
Table 120 – Structure of Connection_Bind_RequestPDU body 2096
Table 121 – Object specific status for Connection_Bind service 2097
Table 122 – Structure of Producing_Application_Lookup_RequestPDU body 2097
Table 123 – Structure of Producing_Application_Lookup_ResponsePDU body 2097
Table 124 – Producing_Application_Lookup Service status codes 2097
Table 125 – Possible port segment examples 2099
Table 126 – TCP/IP link address examples 2100
Table 127 – Electronic key segment format 2102
Table 128 – Logical segments examples 2103
Table 129 – Network segments 2103
Table 130 – Extended subtype definitions 2105
Table 131 – Data segment 2105
Table 132 – ANSI_Extended_Symbol segment 2106
Table 133 – Addressing categories 2106
Table 134 – Class code ID ranges 2107
Table 135 – Attribute ID ranges 2107
Table 136 – Service code ranges 2107
Table 137 – Class codes 2108
Table 138 – Reserved class attributes for all object class definitions 2108
Table 139 – Common services list 2109
Table 140 – Acknowledge Handler object specific services list 2110
Table 141 – Parameter object specific services list 2110
Table 142 – Services specific to Connection Manager 2110
Table 143 – Services specific to Connection object 2110
Table 144 – Device type numbering 2111
Table 145 – Connection Manager service request error codes 2112
Table 146 – General status codes 2117
Table 147 – Identity object status codes 2119
Table 148 – Encapsulation header 2126
Table 149 – Encapsulation command codes 2127
Table 150 – Encapsulation status codes 2128
Table 151 – Options flags 2128
Table 152 – Nop request encapsulation header 2128
Table 153 – RegisterSession request encapsulation header 2129
Table 154 – RegisterSession request data portion 2129
Table 155 – Options flags 2129
Table 156 – RegisterSession reply encapsulation header 2129

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Table 157 – RegisterSession reply data portion 2130
Table 158 – UnRegisterSession request encapsulation header 2130
Table 159 – ListServices request encapsulation header 2131
Table 160 – ListServices reply encapsulation header 2131
Table 161 – ListServices reply data portion 2131
Table 162 – Service type codes 2132
Table 163 – Communications capability flags 2132
Table 164 – ListIdentity request encapsulation header 2132
Table 165 – ListIdentity reply encapsulation header 2133
Table 166 – ListIdentity reply data portion 2133
Table 167 – ListInterfaces request encapsulation header 2134
Table 168 – ListInterfaces reply encapsulation header 2134
Table 169 – SendRRData request encapsulation header 2135
Table 170 – SendRRData request data portion 2135
Table 171 – SendRRData reply encapsulation header 2135
Table 172 – SendUnitData request encapsulation header 2136
Table 173 – SendUnitData request data portion 2136
Table 174 – Common packet format 2136
Table 175 – Address and data item structure 2137
Table 176 – Address type ID's 2137
Table 177 – Data type ID's 2137
Table 178 – Null address type 2137
Table 179 – Connected address type 2138
Table 180 – Sequenced address type 2138
Table 181 – UCMM data type 2138
Table 182 – Connected data type 2138
Table 183 – Sockaddr info items 2139
Table 184 – BOOLEAN encoding 2140
Table 185 – Example compact encoding of a BOOL value 2140
Table 186 – Encoding of SignedInteger values 2140
Table 187 – Example compact encoding of a SignedInteger value 2140
Table 188 – UnsignedInteger values 2141
Table 189 – Example compact encoding of an UnsignedInteger 2141
Table 190 – FixedLengthReal values 2141
Table 191 – Example compact encoding of a REAL value 2141
Table 192 – Example compact encoding of a LREAL value 2141
Table 193 – FixedLengthReal values 2142
Table 194 – STRING value 2142
Table 195 – STRING2 value 2142
Table 196 – STRINGN value 2142
Table 197 – SHORT_STRING value 2142
Table 198 – Example compact encoding of a STRING value 2143
Table 199 – Example compact encoding of STRING2 value 2143

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Table 200 – SHORT_STRING type 2143
Table 201 – Example compact encoding of a single dimensional ARRAY 2145
Table 202 – Example compact encoding of a multi-dimensional ARRAY 2145
Table 203 – Example compact encoding of a STRUCTURE 2146
Table 204 – Identification codes and descriptions of elementary data types 2147
Table 205 – Example 1 of formal encoding of a structure type specification 2148
Table 206 – I/O Connection state event matrix 2154
Table 207 – Bridged Connection state event matrix 2158
Table 208 – Explicit Messaging Connection state event matrix 2160
Table 209 – Primitives issued by FAL user to FSPM 2163
Table 210 – Primitives issued by FAL user to FSPM 2164
Table 211 – Primitives issued by FSPM to FAL user 2166
Table 212 – Parameters used with primitives exchanged between FAL user and FSPM 2167
Table 213 – Primitives issued by FSPM to ARPM 2169
Table 214 – Primitives issued by ARPM to FSPM 2169
Table 215 – Parameters used with primitives exchanged between FSPM and ARPM 2170
Table 216 – UCMM client states 2170
Table 217 – State event matrix of UCMM client 2171
Table 218 – High-end UCMM server states 2172
Table 219 – State event matrix of high-end UCMM server 2174
Table 220 – Low-end UCMM server states 2175
Table 221 – State event matrix of low–end UCMM server 2175
Table 222 – Notification 2178
Table 223 – Transport classes 2179
Table 224 – Primitives issued by FSPM to ARPM 2179
Table 225 – Primitives issued by ARPM to FSPM 2180
Table 226 – Parameters used with primitives exchanged between FSPM and ARPM 2180
Table 227 – Class 0 transport client states 2182
Table 228 – Class 0 client SEM 2182
Table 229 – Class 0 transport server states 2183
Table 230 – Class 0 server SEM 2183
Table 231 – Class 1 transport client states 2186
Table 232 – Class 1 client SEM 2187
Table 233 – Class 1 transport server states 2188
Table 234 – Class 1 server SEM 2189
Table 235 – Class 2 transport client states 2193
Table 236 – Class 2 client SEM 2194
Table 237 – Class 2 transport server states 2195
Table 238 – Class 2 server SEM 2196
Table 239 – Class 3 transport client states 2201
Table 240 – Class 3 client SEM 2202
Table 241 – Class 3 transport server states 2204
Table 242 – Class 3 server SEM 2206

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Table 243 – Write and trigger events in class 4 and 5 transport 2208
Table 244 – Common states for transport classes 4 to 6 2213
Table 245 – Classes 4 to 6 general SEM 2214
Table 246 – Class 4 transport sender states 2216
Table 247 – Class 4 sender SEM 2217
Table 248 – Class 4 transport receiver states 2218
Table 249 – Class 4 receiver SEM 2220
Table 250 – Class 5 transport sender states 2225
Table 251 – Class 5 sender SEM 2226
Table 252 – Class 5 transport receiver states 2228
Table 253 – Class 5 receiver SEM 2230
Table 254 – Class 6 transport client states 2241
Table 255 – Class 6 client state event matrix 2242
Table 256 – Class 6 transport server states 2243
Table 257 – Class 6 server SEM 2245
Table 258 – Primitives issued by ARPM to DMPM 2250
Table 259 – Primitives issued by DMPM to ARPM 2250
Table 260 – Parameters used with primitives exchanged between ARPM and DMPM 2250
Table 261 – Primitives exchanged between data-link layer and DMPM 2250
Table 262 – Parameters used with primitives exchanged between DMPM and Data-link ... 2251
Table 263 – Link producer states 2251
Table 264 – State event matrix of link producer 2252
Table 265 – Link consumer states 2252
Table 266 – State event matrix of link consumer 2253
Table 267 – Network Connection ID selection 2254
Table 268 – Example multicast assignments 2258
Table 269 – UDP data format for class 0 and class 1 2259
Table 270 – Default DSCP and IEEE 802.1D mapping for CP 2/ 2263

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

— Entwurf —

– 2013 –

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

FOREWORD

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

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

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

This edition and of the IEC 61158-6 subseries cancel and replace IEC 61158-6:2007. This edition of this part constitutes a technical revision.

This edition of IEC 61158-6 includes the following significant changes from the previous edition:

- Corrections
- Improvements

AVA Documents
Document Sharing Hub

Changes made in this subpart:

– Clause 2 and Bibliography: update of normative and bibliographic references

– subclause 3.4: update of abbreviations

– subclause 4.1.2.1: update list of service request/response PDUs (Time Sync and Parameter ASEs/objects)

– subclause 4.1.8.6: update of Time Sync ASE/object

– new subclause 4.1.8.7: new Parameter ASE/object

– subclause 4.1.9.4: minor update of a common parameter

– subclause 4.1.10.2: update/add object and services codes for Time Sync and Parameter ASEs

– subclause 4.1.11: minor updates of error codes

– new subclause 11.7: new QoS specification

The text of this standard is based on the following documents:

FDIS	Report on voting
65C/476/FDIS	65C/487/RVD

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

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

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

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

NOTE The revision of this standard will be synchronized with the other parts of the IEC 61158 series.

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

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

— Entwurf —

– 2015 –

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 protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

- as a guide for implementors and designers;
- for use in the testing and procurement of equipment;
- as part of an agreement for the admittance of systems into the open systems environment;
- as a refinement to the understanding of time-critical communications within OSI.

This standard is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this standard together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems may work together in any combination.

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

– 2016 –

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

1 Scope

1.1 General

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

This 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 2 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

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

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

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

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

This standard specify the protocol 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).

1.2 Specifications

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

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of protocols standardized in IEC 61158-6.

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.

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Conformance is achieved through implementation of this application layer protocol specification.

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

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

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

IEC 61588:2008[1], *Precision clock synchronization protocol for networked measurement and control systems*

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

IEC 62026-3[2], *Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet*

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

ISO/IEC 8824, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

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

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

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

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

ISO 11898:1993[3], *Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication*

IEEE 802.1D, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – IEEE standard for local and metropolitan area networks – Common specifications – Media access control (MAC) Bridges*

---

[1] Compliance with future editions of this standard will need checking.

[2] To be published.

[3] A newer edition of this standard has been published, but only the cited edition applies.

AVA Documents Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

IEEE 802.1Q, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – IEEE standard for Local and metropolitan area networks – Virtual bridged local area networks*

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 Terms and definitions from ISO/IEC 7498-1

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

3.1.2 Terms and definitions from ISO/IEC 9545

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

3.1.3 Terms and definitions from ISO/IEC 8824

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

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

3.1.4 Terms and definitions from ISO/IEC 8825

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

3.2 Terms and definitions from IEC 61158-5-2

a) application relationship
b) client
c) peer
o) server

3.3 Additional terms and definitions

3.3.1
allocate
take a resource from a common area and assign that resource for the exclusive use of a specific entity

3.3.2
application
function or data structure for which data is consumed or produced

3.3.3
application objects
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

3.3.4
attribute
description of an externally visible characteristic or feature of an object

NOTE 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.

3.3.5
behaviour
indication of how an object responds to particular events

3.3.6
boundary clock
clock with more than a single PTP port, with each PTP port providing access to a separate PTP communication path

NOTE Boundary clocks are used to eliminate fluctuations produced by routers and similar network elements.

3.3.7
called
service user or a service provider that receives an indication primitive or a request APDU

3.3.8
calling
service user or a service provider that initiates a request primitive or a request APDU

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

3.3.9
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.

3.3.10
class attributes
attribute that is shared by all objects within the same class

3.3.11
class code
unique identifier assigned to each object class

3.3.12
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service

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

3.3.13
client
a) object which uses the services of another (server) object to perform a task
b) initiator of a message to which a server reacts

3.3.14
clock
device providing a measurement of the passage of time since a defined epoch

NOTE There are two types of clocks in IEC 61588:2008, boundary clocks and ordinary clocks.

3.3.15
communication objects
components that manage and provide a run time exchange of messages across the network

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

3.3.16
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.

3.3.17
connection ID (CID)
identifier assigned to a transmission that is associated with a particular connection between producers and consumers, providing a name for a specific piece of application information

3.3.18
connection path
an octet stream which defines the application object to which a connection instance applies

3.3.19
connection point
buffer which is represented as a subinstance of an Assembly object

3.3.20
consume
act of receiving data from a producer

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

3.3.21
consumer
node or sink that is receiving data from a producer

3.3.22
consuming application
application that consumes data

3.3.23
cyclic
repetitive in a regular manner

3.3.24
device
physical hardware connected to the link

NOTE A device may contain more than one node.

3.3.25
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.3.26
end node
producing or consuming node

3.3.27
end point
one of the communicating entities involved in a connection

3.3.28
epoch
reference time defining the origin of a time scale
[IEC 61588:2008]

3.3.29
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.3.30
frame
denigrated synonym for DLPDU

3.3.31
grandmaster clock
clock which serve as the primary source of time to which all others (within a collection of IEC 61588 clocks) are ultimately synchronized.

3.3.32
instance
the actual physical occurrence of an object within a class, identifying one of many objects within the same object class.

EXAMPLE California is an instance of the object class state.

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

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

3.3.33
instance attribute
attribute that is unique to an object instance and not shared by the object class

3.3.34
instantiated
object that has been created in a device

3.3.35
interoperability
capability of User Layer entities to perform coordinated and cooperative operations using the services of the FAL

3.3.36
Keeper
object responsible for distributing link configuration data to all nodes on the link

3.3.37
little endian
Describes a model of memory organisation which stores the least significant octet at the lowest address, or for transfer, which transfers the lowest order octet first

3.3.38
Lpacket
a piece of application information that contains a size, control octet, tag, and link data.

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

3.3.39
management information
network accessible information that supports the management of the Fieldbus environment

3.3.40
master clock
single clock which serves as the primary source of time within each region, and which will in turn synchronize to other master clocks and ultimately to the grandmaster clock

NOTE A system of IEC 61588 clocks may be segmented into regions separated by boundary clocks.

3.3.41
member
piece of an attribute that is structured as an element of an array

3.3.42
Message Router
object within a node that distributes messaging requests to appropriate application objects

3.3.43
multipoint connection
connection from one node to many

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

3.3.44
network
a set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

3.3.45
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

3.3.46
object specific service
service unique to the object class which defines it

3.3.47
ordinary clock
IEC 61588 clock with a single PTP port

3.3.48
originator
client responsible for establishing a connection path to the target

3.3.49
point-to-point connection
connection that exists between exactly two application objects

3.3.50
Precision Time Protocol (PTP)
name used for the time synchronization protocol.

3.3.51
produce
act of sending data to be received by a consumer

3.3.52
producer
node that is responsible for sending data

3.3.53
PTP message
IEC 61588 time synchronization message

NOTE There are five designated messages types defined by IEC 61588:2008: Sync, Delay_Req, Follow-up, Delay_Resp, and Management.

3.3.54
PTP port
logical access point for IEC 61588 communications to the clock containing the port

3.3.55
receiving
service user that receives a confirmed primitive or an unconfirmed primitive, or a service provider that receives a confirmed APDU or an unconfirmed APDU

3.3.56
resource
resource is a processing or information capability of a subsystem

3.3.57
sending
service user that sends a confirmed primitive or an unconfirmed primitive, or a service provider that sends a confirmed APDU or an unconfirmed APDU

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

3.3.58
server
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request
b) object which provides services to another (client) object

3.3.59
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.3.60
synchronized clocks
(to a specified uncertainty) two clocks which have the same epoch and for which measurements of any time interval by both clocks differ by no more than the specified uncertainty

NOTE The timestamps generated by two synchronized clocks for the same event will differ by no more than the specified uncertainty.

3.3.61
System Time
absolute time value as defined by CPF2 time synchronization in the context of a distributed time system where all devices have a local clock that is synchronized with a common master clock

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

3.3.62
target
end-node to which a connection is established

3.3.63
temporary node
transient node

3.3.64
transaction id
field within a UCMM header that matches a response with the associated request

3.3.65
Unconnected Message Manager (UCMM)
component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object

3.3.66
unconnected service
messaging service which does not rely on the set up of a connection between devices before allowing information exchanges

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

3.4 Abbreviations and symbols

ASCII	American Standard Code for Information Interchange
CID	connection ID
DLL	data-link layer
DSCP	DiffServ Codepoint
IGMP	Internet Group Management Protocol (see RFC 1112, RFC 2236)
IP	Internet Protocol (see RFC 791)
IPv4	Internet Protocol version 4 (see RFC 791)
IPv6	Internet Protocol version 6 (see RFC 791)
OSI	open systems interconnection (see ISO/IEC 7498 series)
PDU	protocol data unit
PHB	per-hop behavior
PTP	Precision Time Protocol [IEC 61588:2008]
QoS	quality of service
Rcv	receive
Rx	receive
SDU	service data unit
SEM	state event matrix
STD	state transition diagram, used to describe object behaviour
TCP	Terminal Control Protocol (see RFC 793)
ToS	type of service
TPDU	transport protocol data unit
Tx	transmit
UDP	User Datagram Protocol (see RFC 768)
Xmit	transmit
CM_API	actual packet interval
O2T or O⇒T	originator to target (connection parameters)
CM_RPI	requested packet interval
TUI	table unique identifier
T2O or T⇒O	target to originator (connection parameters)

3.5 Conventions

3.5.1 General concept

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

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in IEC 61158-5-2. The service specification defines the services that are provided by the ASE.

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

3.5.1.1 Attribute specification

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

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Attribute ID	Name	Data type	Semantics of values

Figure 1 – Attribute table format and terms

The **Attribute ID** shall be a unique integer identification value assigned to an attribute. The valid ranges for Attribute IDs shall be as specified in 5.1.1.10.1.3 The Attribute ID shall identify the particular attribute being accessed.

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

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

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

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

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

3.5.1.2 Common services

3.5.1.2.1 Service_PDU definitions

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

Name	Type	Semantics of values

Figure 2 – Service request/response parameter

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

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

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

3.5.1.2.2 Get_Attribute_All response

3.5.1.2.2.1 General definition

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

- list the ordering of the attributes in the response message;
- specify the actual data array of the response message;

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

- combine the above two methods such that the actual data array also contains the attribute numbers.

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

Table 1 – Get_Attribute_All response service rules

Rule number	Rule
1	If the definition of the Get_Attribute_All response includes optional attributes, then default values shall be specified in the response description. These defaults shall be used if an implementation does not support the optional attributes. If any of the following optional attributes are included in an object specification, but not supported in the implementation of the object, then the following shall be adhered to: – if the class attribute “Optional attribute list” is not supported, the default value of zero shall be inserted into the response array and no optional attribute numbers shall follow; – if the class attribute “optional service list” is not supported, the default value of zero shall be inserted into the response array and no optional service numbers shall follow.
2	If new attributes are added to an existing object, those attributes shall be added to the end of the response attribute list or data array to ensure compatibility with different object revisions.
3	Whichever method is used to specify the response, it shall be done in such a way as to be unambiguous, including rules to deal with variable length fields and padding.
4	The Get_Attribute_All response shall include only the open attributes; it shall not include any vendor specific attributes.

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

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

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

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

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Table 3 – Example Get_Attribute_All data array method

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

3.5.1.2.2.2 Revisions

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

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

3.5.1.2.3 Set_Attribute_All request

3.5.1.2.3.1 General definition

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

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

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

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Table 4 – Set_Attribute_All request service rules

Rule number	Rule
1	An object shall support the Set_Attribute_All service only if all settable attributes shown in the Set_Attribute_All request are implemented as settable.
2	If new settable attributes are added to an existing object, those attributes shall be added to the end of the request attribute list or data array.
3	Whichever method is used to specify the response, it shall be done in such a way as to be unambiguous, including rules to deal with variable length fields and padding.
4	The Set_Attribute_All request shall include only the open attributes. It shall not include any vendor specific attributes.

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

Table 5 – Example Set_Attribute_All attribute ordering method

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

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

Table 6 – Example Set_Attribute_All data array method

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

3.5.1.2.3.2 Revisions

A server processing a Set_Attribute_All request may need to process more data than is expected by the server if the client has implemented a newer revision of this object while the server an older revision. As a result, the server object may have to process a Set_Attribute_All request that contains more data because the additional attributes were not recognised. If the server receives more data in a Set_Attribute_All request than it expects the server shall respond with a general status code equal to *Too much data* (0x15).

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

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

3.6 Conventions used in state machines

3.6.1 State machine conventions for Type 2

3.6.1.1 General

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

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

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

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

3.6.1.2 State Transition Diagram (STD)

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

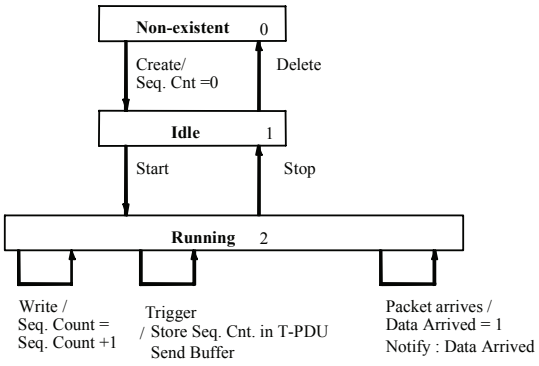

Figure 3 – Example of an STD

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

3.6.1.3 State Event Matrix

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

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

State event matrix format is as shown in Table 7.

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Table 7 – State event matrix format

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

3.6.1.4 Example state event matrix

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

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

Table 8 – Example state event matrix

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

4 Abstract syntax

4.1 FAL PDU abstract syntax

4.1.1 General

FAL_PDU shall be either a UCMM_PDU or a Transport_PDU.

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

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

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

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

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

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

4.1.2 PDU structure

4.1.2.1 UCMM_PDU structure

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

OM_Service shall be either OM_Request or OM_Response.

OM_Request shall consist of MR_Request_Header and a Service_RequestPDU.

Service_RequestPDU shall be one of the following :

- Get_Attribute_Single_RequestPDU
- Get_Attribute_All_RequestPDU
- Get_Attribute_List_RequestPDU
- Set_Attribute_Single_RequestPDU
- Set_Attribute_All_RequestPDU
- Set_Attribute_List_RequestPDU
- Reset_RequestPDU
- Create_RequestPDU
- Delete_RequestPDU
- Start_RequestPDU
- Stop_RequestPDU
- Find_Next_Object_Instance_RequestPDU
- NOP_RequestPDU
- Apply_Attributes_RequestPDU
- Save_RequestPDU

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

- Restore_RequestPDU
- Group_Sync_RequestPDU
- Add_AckData_Path_RequestPDU
- Remove_AckData_Path_RequestPDU
- Get_Enum_String_RequestPDU
- Connection_Bind_RequestPDU
- Producing_Application_Lookup_RequestPDU

OM_Response shall consist of MR_Response_Header and a Service_ResponsePDU.

Service_ResponsePDU shall be one of the following :

- Get_Attribute_Single_ResponsePDU
- Get_Attribute_All_ResponsePDU
- Get_Attribute_List_ResponsePDU
- Set_Attribute_Single_ResponsePDU
- Set_Attribute_All_ResponsePDU
- Set_Attribute_List_ResponsePDU
- Reset_ResponsePDU
- Create_ResponsePDU
- Delete_ResponsePDU
- Start_ResponsePDU
- Stop_ResponsePDU
- Find_Next_Object_Instance_ResponsePDU
- NOP_ResponsePDU
- Apply_Attributes_ResponsePDU
- Save_ResponsePDU
- Restore_ResponsePDU
- Group_Sync_ResponsePDU
- Add_AckData_Path_ResponsePDU
- Remove_AckData_Path_ResponsePDU
- Get_Enum_String_ResponsePDU
- Connection_Bind_ResponsePDU
- Producing_Application_Lookup_ResponsePDU

CM_Service shall be either CM_Request or CM_Response.

CM_Request shall consist of MR_Request_Header and a CM_RequestPDU.

CM_RequestPDU shall be one of the following :

- Forward_Open_RequestPDU
- Forward_Close_RequestPDU
- Large_Forward _Open_RequestPDU
- Unconnected_Send_RequestPDU
- Get_Connection_Data_RequestPDU

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

- Search_Connection_Data _RequestPDU
- Get_Object_Owner_RequestPDU

CM_Response consists of MR_Response_Header and a CM_ResponsePDU.

CM_ ResponsePDU shall be one of the following :

- Forward_Open_ ResponsePDU
- Forward_Close_ ResponsePDU
- Large_Forward _Open_ ResponsePDU
- Unconnected_Send_ ResponsePDU
- Get_Connection_Data_ ResponsePDU
- Search_Connection_Data _ ResponsePDU
- Get_Object_Owner_ ResponsePDU

4.1.2.2 Transport_PDU structure

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

OM_Service is defined in 4.1.2.1 above.

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

4.1.3 UCMM_PDUs

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

Table 9 – UCMM_PDU header format

Parameter name	Format
command_code	USINT
Timeout	USINT
TransactionID	UINT

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

Table 10 – UCMM command codes

Command_code	Description
0	Reserved
1	acknowledge a request
2	request with retry until acknowledged
3	response with retry until acknowledged

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Command_code	Description
4	request with no acknowledge and no response
5	acknowledge a response
6	response which shall not retry (no acknowledge)
7	request with retry until response (no acknowledge)
8	Request which shall not retry and shall cause a code 6 response
9 – 255	Reserved

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

NOTE Using ANSI C precedence rules, the number of 0,125 ms ticks is
(8 | `timeout` & 7) << (`timeout` >> 3 & 31) .

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

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

Only one message shall be outstanding for a given RECORD.

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

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

4.1.4 Transport_Headers

4.1.4.1 General

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

4.1.4.2 Class 0 (null or base)

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

Table 11 – Transport class 0 header

Parameter name	Format
don't_care	UINT

4.1.4.3 Class 1 (duplicate detection)

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

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Table 12 – Transport class 1 header

Parameter name	Format
sequence_count	UINT

4.1.4.4 Class 2 (acknowledged)

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

Table 13 – Transport class 2 header

Parameter name	Format
sequence_count	UINT

4.1.4.5 Class 3 (verified)

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

Table 14 – Transport class 3 header

Parameter name	Format
sequence_count	UINT

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

These classes of transports all use the header format shown in Table 15. Classes 4 and 5 have two headers in each TPDU, one Sender and one Receiver. Class 6 uses a Sender header in the Client to Server direction and a Receiver header in the Server to Client direction.

Table 15 – Classes 4 to 6 header format

Command			Sequence number	Data	
MSB	Bits 15 – 12		Bits 11 – 8	Bits 7 – 0	LSB
Value:	Meaning in Sender Header:	Meaning in Receiver Header:	Increments from 0 to 15, then rolls over to zero again. Used in Sender header to distinguish a retry from a new transmission. Used in Receiver header to indicate which sender TPDU is being ACK'd or NAK'd.	For "First": Number of 256 octet blocks (minus 1) to allocate to hold complete message. For "NAK": Reason code. For others: Set to zero when generating; ignore when receiving.	
0.	Null	ACK			
1.	Only	NAK			
2.	First	Not Initialised			
3.	Middle	Undefined			
4.	Last	Undefined			

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Command		Sequence number	Data	
MSB	Bits 15 – 12	Bits 11 – 8	Bits 7 – 0	LSB
5. to 15 are undefined.				

The Command identifies the use of the TPDU by the transport. The Command field is defined differently for the Sender and the Receiver header.

Sender header commands :

Null = no data in this Transport PDU

Only = this is the Transport PDU conveying all of the Application data

First / Middle / Last = position of this Transport PDU within the segmented Application data

The Sequence Number is used in the Sender header to distinguish a retry from a new transmission. It is used in Receiver header to indicate which sender TPDU is being ACK'd or NAK'd. This is **NOT** used for applications request/response matching. This is only used by the transport state machines.

The Data is used by one Sender command and one Receiver command. It is used in the *First* Sender command to indicate the number of 256 octet blocks (minus 1) the Receiver shall allocate for the complete application message. That is, application messages of size 0 to 256 octets shall have the *First* Data field set to 0; 257 to 512, to 1; and so on up to 65 536 octets. It is used in the NAK Receiver command to provide a reason code:

Class 4 (the Point-to-Point non-Blocking transport) uses two headers in each TPDU, followed by the application data. The Sender Header is first in the TPDU, the Receiver Header is second, followed by the application data. Class 4 only uses the *Null* and *Only* Sender Commands (numbered 0 and 1).

Class 5 (the Point-to-Point Fragmenting non-Blocking transport) uses two headers in each TPDU, followed by the application data. The Sender Header is first in the TPDU, the Receiver Header is second, followed by the application data. Class 5 uses all of the Sender Commands.

Class 6 (the Multipoint connection Fragmenting transport) uses one header in each TPDU, followed by the application data. Class 6 uses all of the Sender Commands. The Sender Header is used in the Client-to-Server TPDU and the Receiver Header is used in the Server to Client TPDU.

4.1.4.7 Exclusive owner O⇒T data format

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

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

The 32-bit header prefixed to the real-time data shall be in the form shown in Table 16.

Table 16 – Real-time data header – exclusive owner

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

AVA Documents Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

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

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

4.1.4.8 Redundant owner O⇒T data format

4.1.4.8.1 General format

Redundant owner shall have a 32-bit header prefixed to the real-time data. This header shall be in the form shown in Table 17.

Table 17 – Real-time data header– redundant owner

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

4.1.4.8.2 Run_idle flag

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

4.1.4.8.3 Claim output ownership (COO) flag

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

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

4.1.4.8.4 Ready for ownership of outputs (ROO) priority value

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

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

4.1.4.8.5 Determining the owning connection

The originator applications shall determine among themselves which originator application has the owning connection. The owning connection shall be determined by the originator application that sets its COO flag. In situations where multiple originator applications have

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

their COO flag set or where no originator connections have their COO flag set, the following rules shall be applied by the target transport to determine the owning connection.

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

4.1.4.8.6 Transporting events and data to a target application

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

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

4.1.5 CM_PDUs

4.1.5.1 General

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

CM_Forward_Open (corresponds to Forward_Open and Ex_ Forward_Open PDUs)

CM_Forward_Close (corresponds to Forward_Close PDUs)

CM_Unconnected_Send (corresponds to Unconnected_Send PDUs)

CM_Get_Connection_Data (corresponds to Get_Connection_Data PDUs)

CM_Search_Connection_Data (corresponds to Search_Connection_Data PDUs)

CM_Get_Object_Owner (corresponds to Get_Object_ Owner PDUs)

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

4.1.5.2 Connection Manager

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

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

4.1.5.3 Forward_Open

4.1.5.3.1 General

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

NOTE 1 The Forward_Open request sets up network, transport, and application connections. An application connection consists of a single transport connection, and one or two network connections that are in turn comprised of multiple link connections. Each port segment in the connection path uses a link connection. The Forward_Open service between two devices builds one or two link connections as specified by the network connection parameter and the requested packet intervals (CM_RPI). Since up to two network connections can be required for a single transport connection, they are differentiated by the O⇒T and T⇒O designations; O⇒T means originator to target, and T⇒O means target to originator.

NOTE 2 The processing of a single Forward Open service may result in the creation of multiple active Connection object instances.

4.1.5.3.2 Format of Forward_Open request

The format of the Forward_Open request TPDU shall be of the form shown in Table 18.

Table 18 – Forward_Open request format

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

4.1.5.3.3 Format of Forward_Open response if success

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

AVA Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Table 19 – Forward_Open_Good response format

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

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

4.1.5.3.4 Format of Forward_Open response if failure

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

Table 20 – Forward_Open_Bad response format

Parameter name	Format
connection_serial_number	UINT
vendor_ID	UINT
originator_serial_number	UDINT
remaining_path_size; (in 16-bit words)	USINT
Reserved	USINT

This format shall be used for all failures of the connection system. The requested connection shall not be established, and the object specific status words shall contain information about the reason for the failure. The `remaining_path_size` shall contain the length of the path at the point the connection failed. This information can be used to debug the problem.

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

A duplicate Forward_Open service shall be defined as a Forward_Open service whose `vendor_ID`, `connection_serial_number`, and `originator_serial_number` match an existing connection’s parameters. If the duplicate Forward_Open service is a null Forward_Open service (defined as the connection type in both the O2T and T2O network connection parameter fields are NULL), then the Forward_Open service shall be forwarded to the application for further processing. Null Forward_Open requests may be used to reconfigure the connection. The Connection Manager in the intermediate nodes need not allocate additional resources for a duplicate Forward_Open request since the resources have

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

already been allocated. If the duplicate Forward_Open request is not NULL, then an general status = 0x01, extended status = 0x0100 shall be returned.

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

4.1.5.4 Large_Forward_Open

4.1.5.4.1 General

The Large_Forward_Open shall have the same function as the Forward_Open except that it shall allow the establishment of connections larger than 511 octets.

The response to the Large_Forward_Open_Request_PDU is an Large_Forward_Open_ResponsePDU. The Large_Forward_Open response shall contain the original connection serial number and owner vendor and serial number and the connection IDs (CID) necessary to complete the connection if successful and error information if the connection failed.

NOTE The Large_Forward_Open request is very similar to the Forward_Open request.

4.1.5.4.2 Format of Large_Forward_Open request

The format of the Large_Forward_Open request TPDU shall be of the form shown in Table 21.

Table 21 – Large_Forward_Open request format

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

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

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

4.1.5.4.3 Format of Large_Forward_Open response if success

If the `status` parameter of the `CM_open_response` is zero (no error), the format of the Large_Forward_Open response TPDU shall be of the form shown in Table 22.

Table 22 – Large_Forward_Open_Good response format

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

Success shall be returned when the connection requested has been established from this point forward in the path. This response also shall indicate the connection serial number and the actual packet rate of the connection. Once the successful response has been received, the connection shall be open from this point forward in the path.

4.1.5.4.4 Format of Large_Forward_Open response if failure

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

Table 23 – Large_Forward_Open_Bad response format

Parameter name	Format
connection_serial_number	UINT
vendor_ID	UINT
originator_serial_number	UDINT
remaining_path_size; (in 16-bit words)	USINT
Reserved	USINT

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

This format shall be used for all failures of the connection system. The requested connection shall not be established, and the object specific status words shall contain information about the reason for the failure. The `remaining_path_size` shall contain the length of the path at the point the connection failed. This information can be used to debug the problem.

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

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

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

4.1.5.5 Forward_Close

4.1.5.5.1 General

The Forward_Close request shall remove a connection from all the nodes participating in the original connection. The Forward_Close shall be sent between Connection Managers as specified in the connection_path. The Forward_Close request shall cause all resources in all nodes participating in the connection to be deallocated, including connection IDs, link transmit time, and internal memory buffers.

If an intermediate node cannot find the connection that is to be closed (it may have timed out at the node), the Forward_Close request shall still be forwarded to downstream nodes or the target application (via the `CM_close_indication`).

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

4.1.5.5.2 Format of Forward_Close request

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

Table 24 – Forward_Close request format

Parameter name	Format
connection_priority/tick time	SWORD
connection_timeout	USINT
connection_serial_number	UINT
vendor_ID	UINT
originator_serial_number	UDINT
connection_path_size	USINT
Reserved	USINT
connection_path	Padded EPATH

4.1.5.5.3 Format of Forward_Close response if success

The Forward_Close response shall be sent between Connection Managers in response to a Forward_Close request, and shall contain the status of the close request. If the `status` parameter of the `CM_close_response` is zero (no error), the format of the Forward_Close response TPDU shall be of the form shown in Table 25.

Table 25 – Forward_Close_Good response format

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

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

A successful Forward_Close response shall be returned when the close has been acknowledged by all the nodes from this point in the path to the end of the path. The response shall indicate the `connection_serial_number`, `vendor_ID`, and `originator_serial_number` of the closed connection. Once the response indicating success has been received, the connection shall be closed from this point in the path to the end. The target application may optionally pass back application specific information in the successful close response. If no `application_reply` information is contained in the service the `application_reply_size` shall be zero.

4.1.5.5.4 Format of Forward_Close response if failure

If the `status` parameter of the `CM_close_response` is not zero (error), the format of the Forward_Close response shall be of the form shown in Table 26.

Table 26 – Forward_Close_Bad response format

Parameter name	Format
connection_serial_number	UINT
vendor_ID	UINT
originator_serial_number	UDINT
remaining_path_size	USINT
reserved	USINT

The object specific status words shall contain information about the reason for the failure. The `remaining_path_size` shall contain the path size from the point at which the close connection failed.

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

4.1.5.6 Unconnected_Send

4.1.5.6.1 Format of Unconnected_Send request

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

NOTE The target node never sees the Unconnected_Send service but only a standard message arriving via the UCMM.

If the message contains an odd number of octets, a pad octet shall be appended to the end of the message allowing the remaining fields to be aligned on a 16-bit boundary. The format of the Unconnected_Send request shall be of the form of the form shown in Table 27.

AVA Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06

Table 27 – Unconnected_Send request format

Parameter name	Format
priority/tick time	SWORD
conn time-out ticks	USINT
Message_size; in octets, not including pad	UINT
Message Router PDU ; if message_router_PDU is odd size, include pad octet	MR_Request
pad	USINT
path_size	USINT
Reserved	USINT
path	Padded EPATH

The `message_size` shall be the size of the `message_router_PDU` in octets without the `pad`. The `reserved` field shall be set to zero. The `path_size` shall be the number of 16-bit words in the `path` field.

4.1.5.6.2 Unconnected_Send response

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

4.1.5.6.3 Format of Unconnected_Send response if success

This format shall be used when an unconnected message response is received. The application response shall be the format of the Message response from the target and shall contain error codes that resulted from the execution of the message by the application at the target node. The structure of the response is shown in Table 28.

Table 28 – Unconnected_Send_Good response format

Parameter name	Format
application_reply [ ]	OCTET

4.1.5.6.4 Format of Unconnected_Send response if failure

This format shall be used if the originating node or any intermediate node found a problem in the message format, message parameters, or a UCMM timed-out due to no acknowledge or no response received. A UCMM time-out may be the result of

- the target device not being connected or powered up. (link specific time-out);
- an intermediate device not being connected or powered up. (link specific time-out);
- a target device being out of UCMM buffers or overloaded (Flow Control). (link specific time-out);
- an intermediate device being out of UCMM buffers or overloaded (Flow Control). (link specific time-out);
- an invalid path segment type (logical or network segment);
- an invalid port segment address (link or port address invalid).

Documents
Document Sharing Hub
Full document details: E DIN IEC 61158-600:2009-06

Since the Unconnected message system does not implement any flow control, the failure of an Unconnected_Send message need not mean a device is not present in the system. The format of the Unconnected_Send response service for a failure shall be of the form shown in Table 29.

Table 29 – Unconnected_Send_Bad response format

Parameter name	Format
remaining_path_size	USINT
pad	USINT

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

4.1.5.7 Get_Connection_Data

4.1.5.7.1 General

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

NOTE This service may be used for network diagnostics.

4.1.5.7.2 Format of Get_Connection_Data request

The format of the Get_Connection_Data request shall be of the form shown in Table 30.

Table 30 – Get_Connection_Data request format

Parameter name	Format
connection_number	UINT

4.1.5.7.3 Format of Get_Connection_Data response

The format of the Get_Connection_Data response TPDU shall be of the form shown in Table 31.

Documents
Document Sharing Hub

Full document details: E DIN IEC 61158-600:2009-06