

**DIN EN 61158-6-15**

ICS 35.100.70; 35.160

Ersatz für  
DIN EN 61158-6-15:2008-09  
Siehe Anwendungsbeginn

**Industrielle Kommunikationsnetze –  
Feldbusse –  
Teil 6-15: Protokollspezifikation des Application Layer  
(Anwendungsschicht) – Typ 15-Elemente  
(IEC 61158-6-15:2010);  
Englische Fassung EN 61158-6-15:2012, nur auf CD-ROM**

Industrial communication networks –  
Fieldbus specifications –  
Part 6-15: Application layer protocol specification – Type 15 elements  
(IEC 61158-6-15:2010);  
English version EN 61158-6-15:2012, only on CD-ROM

Réseaux de communication industriels –  
Spécifications des bus de terrain –  
Partie 6-15: Spécification des protocoles des couches d'application – Eléments de type 15  
(CEI 61158-6-15:2010);  
Version anglaise EN 61158-6-15:2012, seulement en CD-ROM

Gesamtumfang 108 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

## Anwendungsbeginn

Anwendungsbeginn für die von CENELEC am 2012-03-28 angenommene Europäische Norm als DIN-Norm ist 2013-03-01.

Für DIN EN 61158-6-15:2008-09 gilt eine Übergangsfrist bis zum 2015-03-28.

## Nationales Vorwort

*Vorausgegangener Norm-Entwurf: E DIN IEC 61158-600:2009-06.*

Für dieses Dokument ist das nationale Arbeitsgremium K 956 „Feldbus“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE ([www.dke.de](http://www.dke.de)) zuständig.

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

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

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

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

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

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

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.

Da sich die Benutzer der vorliegenden Norm der englischen Sprache als Fachsprache bedienen, wird die englische Fassung der EN 61158-6-15 veröffentlicht. Zu deren Abschnitt 3, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als informativer Nationaler Anhang NA der vorliegenden Norm hinzugefügt. Für viele der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Diese Norm steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

## 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 Typ 15 fest.

## Änderungen

Gegenüber DIN EN 61158-6-15:2008-09 wurden folgende Änderungen vorgenommen:

- a) redaktionelle Verbesserungen.

**Frühere Ausgaben**

DIN V 19245-1: 1988-01  
 DIN EN 61158-6: 2005-05  
 DIN EN 61158-6-15: 2008-09

## **Nationaler Anhang NA** (informativ)

**3.1.5 Begriffe aus IEC/TR 61158-1**

Es gelten die folgenden Begriffe aus IEC/TR 61158-1.

**3.1.5.1****Anwendung**

(en: **application**)

Funktion oder Datenstruktur, für die Daten verarbeitet oder erzeugt werden

**3.1.5.2****Interoperabilität auf Anwendungsschicht**

(en: **application layer interoperability**)

Fähigkeit von Anwendungseinheiten, mithilfe der Dienste der FAL koordinierte und kooperative Operationen auszuführen

**3.1.5.3****Anwendungsobjekt**

(en: **application object**)

Objektklasse, die den Nachrichtenaustausch über das Netzwerk und innerhalb des Netzwerkgerätes zur Laufzeit bereitstellt und verwaltet

ANMERKUNG Es können mehrere Typen von Anwendungsobjektklassen definiert werden.

**3.1.5.4****Anwendungsprozess**

(en: **application process**)

Teil einer verteilten Anwendung in einem Netzwerk, der sich auf einem Gerät befindet und eindeutig adressiert wird

**3.1.5.5****Anwendungsprozesskennung**

(en: **application process identifier**)

unterscheidet mehrere in einem Gerät verwendete Anwendungsprozesse

**3.1.5.6****Anwendungsprozessobjekt**

(en: **application process object**)

Komponente eines Anwendungsprozesses, die über eine FAL-Anwendungsbeziehung identifizierbar und zugänglich ist

ANMERKUNG Definitionen von Anwendungsprozessobjekten bestehen aus einem Satz von Werten für die Attribute ihrer Klasse.

**3.1.5.7****Klasse von Anwendungsprozessobjekten**

(en: **application process object class**)

Klasse von Anwendungsprozessobjekten, die in Bezug auf den Satz der über das Netzwerk zugänglichen Attribute und Dienste der Anwendungsprozessobjekte definiert ist

### 3.1.5.8

#### **Anwendungsbeziehung**

(en: **application relationship**)

kooperative Verbindung zwischen zwei oder mehr Aufrufen von Anwendungseinheiten zum Zweck des Informationsaustausches und der Koordination des gemeinsamen Betriebs

ANMERKUNG Diese Beziehung wird entweder durch den Austausch von Protokolldateneinheiten der Anwendung oder als Ergebnis von Vorkonfigurationsaktivitäten aktiviert.

### 3.1.5.9

#### **Endpunkt der Anwendungsbeziehung**

(en: **application relationship endpoint**)

Kontext und Verhalten einer Anwendungsbeziehung entsprechend der Sicht und Verwaltung durch einen der in die Anwendungsbeziehung eingeschlossenen Anwendungsprozesse

ANMERKUNG Jeder Anwendungsprozess, der in die Anwendungsbeziehung eingeschlossen ist, verwaltet seinen eigenen Endpunkt der Anwendungsbeziehung.

### 3.1.5.10

#### **Anwendungsdienstelement**

(en: **application service element**)

Anwendungsdienstelement, das die alleinigen Mittel für den Aufbau und die Beendigung aller Anwendungsbeziehungen bereitstellt

### 3.1.5.11

#### **Attribut**

(en: **attribute**)

Beschreibung eines von außen sichtbaren Merkmals oder einer von außen sichtbaren Eigenschaft eines Objekts

ANMERKUNG Die Attribute eines Objekts enthalten Informationen zu veränderlichen Teilen eines Objekts. Üblicherweise stellen sie Statusinformationen bereit oder steuern den Betrieb eines Objekts. Attribute können auch das Verhalten eines Objekts beeinflussen. Attribute werden in Klassenattribute und Instanzattribute unterteilt.

### 3.1.5.12

#### **Verhalten**

(en: **behavior**)

Angabe, wie das Objekt auf bestimmte Ereignisse reagiert

ANMERKUNG Seine Beschreibung schließt die Beziehung zwischen Attributwerten und Diensten ein.

### 3.1.5.13

#### **Klasse**

(en: **class**)

Satz von Objekten, die allesamt den gleichen Typ von Systemkomponente repräsentieren

ANMERKUNG Eine Klasse ist eine Verallgemeinerung des Objektes; eine Vorlage für die Festlegung von Variablen und Methoden. Alle Objekte in einer Klasse sind in ihrer Form und in ihrem Verhalten identisch, enthalten aber üblicherweise unterschiedliche Daten in ihren Attributen.

### 3.1.5.14

#### **Klassenattribute**

(en: **class attributes**)

Attribut, das von allen Objekten innerhalb derselben Klasse gemeinsam genutzt wird

### 3.1.5.15

#### **Klassencode**

(en: **class code**)

eindeutige Kennung, die jeder Objektklasse zugewiesen ist

**3.1.5.16****klassenspezifischer Dienst****(en: class specific service)**

Dienst, der durch eine bestimmte Objektklasse definiert ist, eine erforderliche Funktion, die nicht von einem allgemeinen Dienst ausgeführt wird, auszuführen

ANMERKUNG Ein klassenspezifisches Objekt ist für die Objektklasse, die es definiert, eindeutig.

**3.1.5.17****Client****(en: Client)**

- a) Objekt, das die Dienste eines anderen Objekts (Servers) nutzt, um eine Aufgabe auszuführen
- b) Initiator einer Nachricht, auf die ein Server reagiert, wie die Rolle eines AR-Endpunktes, in der dieser bestätigte Dienstanforderungs-APDUs an einen einzelnen AR-Endpunkt, der als Server agiert, ausgibt

**3.1.5.18****Übertragungspfad****(en: conveyance path)**

unidirektionaler Strom von APDUs über eine Anwendungsbeziehung

**3.1.5.19****zyklisch****(en: cyclic)**

Benennung zur Beschreibung von Ereignissen, die sich regelmäßig wiederholen

**3.1.5.20****zugeordnete AR****(en: dedicated AR)**

AR, die direkt vom FAL-Benutzer verwendet wird

ANMERKUNG Bei zugeordneten Anwendungsbeziehungen werden nur der FAL-Header und die Benutzerdaten übertragen.

**3.1.5.21****Gerät****(en: device)**

physikalische Hardware-Verbindung zum Verbindungszweig

ANMERKUNG Ein Gerät kann mehr als einen Knoten umfassen.

**3.1.5.22****Geräteprofil****(en: device profile)**

Zusammenstellung geräteabhängiger Informationen und Funktionalitäten, um Konsistenz zwischen ähnlichen Geräten desselben Gerätetyps herzustellen

**3.1.5.23****dynamische AR****(en: dynamic AR)**

AR, für die die Nutzung der Prozeduren für die Herstellung von ARs erforderlich ist, um sie in einen aktiven Zustand zu versetzen

**3.1.5.24****Endpunkt****(en: endpoint)**

eine der Kommunikationseinheiten, die zu einer Verbindung gehören

#### 3.1.5.25

##### **Fehler**

(en: error)

Nichtübereinstimmung zwischen einem berechneten, beobachteten oder gemessenen Wert oder Zustand und dem festgelegten oder theoretisch richtigen Wert oder Zustand

#### 3.1.5.26

##### **Fehlerklasse**

(en: error class)

allgemeine Gruppierung für Fehlerdefinitionen

ANMERKUNG Innerhalb einer Fehlerklasse sind Fehlercodes für spezifische Fehler festgelegt.

#### 3.1.5.27

##### **Fehlercode**

(en: error code)

Kennzeichnung eines bestimmten Fehlertyps innerhalb einer Fehlerklasse

#### 3.1.5.28

##### **FAL-Subnetz**

(en: FAL subnet)

Netzwerke, die aus einem oder mehreren Segmenten auf Sicherungsschicht bestehen

ANMERKUNG FAL-Subnetze dürfen Bridges, jedoch keine Router umfassen. FAL-Subnetze werden durch eine Unter-  
menge der Netzwerkadresse identifiziert.

#### 3.1.5.29

##### **logisches Gerät**

(en: logical device)

FAL-Klasse, die eine Softwarekomponente oder eine Firmwarekomponente als eigenständige Einrichtung  
eines Automatisierungsgerätes abstrahiert

#### 3.1.5.30

##### **Verwaltungsinformationen**

(en: management information)

über das Netzwerk zugängliche Informationen, die die Verwaltung des Betriebs des Feldbussystems,  
einschließlich der Anwendungsschicht, unterstützen

ANMERKUNG Zur Verwaltung gehören Funktionen wie die Steuerung, die Überwachung und die Diagnose.

#### 3.1.5.31

##### **Netzwerk**

(en: network)

Reihe von Knoten, die über ein bestimmtes Kommunikationsmedium miteinander verbunden sind

ANMERKUNG Die Verbindungspfade zwischen beliebigen Paaren von Knoten können Repeater, Router und  
Gateways einschließen.

#### 3.1.5.32

##### **Peer**

(en: peer)

Rolle eines AR-Endpunktes, in der dieser in der Lage ist, sowohl als Client als auch als Server zu agieren

#### 3.1.5.33

##### **vordefinierter AR-Endpunkt**

(en: pre-defined AR endpoint)

AR-Endpunkt, der lokal innerhalb eines Gerätes ohne Nutzung des Erstellungsdienstes definiert ist

ANMERKUNG Vordefinierte ARs, die nicht bereits bestehen, werden vor ihrer Nutzung hergestellt.

**3.1.5.34****vorher aktivierter AR-Endpunkt****(en: pre-established AR endpoint)**

AR-Endpunkt, der während der Konfiguration der AEs, die ihre Endpunkte steuern, in einen aktiven Zustand versetzt wird

**3.1.5.35****Publisher****(en: publisher)**

Rolle eines AR-Endpunktes, in der dieser APDUs auf den Feldbus für den Verbrauch durch einen oder mehrere Subscriber überträgt

ANMERKUNG Der Publisher muss die Identität oder Anzahl der Subscriber nicht kennen und kann seine APDUs mithilfe einer zugeordneten AR veröffentlichen. In dieser Norm sind zwei Typen von Publishern festgelegt: Pull-Publisher und Push-Publisher, die jeweils einzeln definiert sind.

**3.1.5.36****Server****(en: server)**

- a) Rolle eines AREP, in der dieser eine bestätigte Dienstantwort-APDU an den Client, der die Anforderung initiiert hat, zurückgibt
- b) Objekt, das einem anderen Objekt (Client) Dienste bereitstellt

**3.1.5.37****Dienst****(en: service)**

Operation oder Funktion, die ein Objekt und/oder eine Objektklasse auf Aufforderung eines anderen Objekts und/oder einer anderen Objektklasse ausführt

ANMERKUNG Es ist eine Reihe von allgemeinen Diensten definiert und es sind Richtlinien für die Definition von objektspezifischen Diensten angegeben. Objektspezifische Dienste sind durch eine bestimmte Objektklasse definierte Dienste, die eine erforderliche Funktion, die nicht von einem allgemeinen Dienst ausgeführt wird, ausführen sollen.

**3.1.5.38****Subscriber****(en: subscriber)**

Rolle eines AREP, in der dieser von einem Publisher produzierte APDUs empfängt

ANMERKUNG In dieser Norm sind zwei Typen von Subscribern festgelegt: Pull-Subscriber und Push-Subscriber, die jeweils einzeln definiert sind.

**3.1.6 Spezielle Begriffe zum Client/Server****3.1.6.1****Coils, diskrete Ausgänge****(en: coils, discrete outputs)**

Anwendungsprozessobjekt; eine Menge von Coils, gekennzeichnet durch die Adresse eines Coil und eine Menge von Coils; diese Menge wird im Zusammenhang mit Felddausgängen auch als „diskrete Ausgänge“ bezeichnet

**3.1.6.2****Discrete Input, diskreter Eingang****(en: discrete, discrete input)**

Anwendungsprozessobjekt; adressiert durch eine vorzeichenlose Zahl und mit einer Breite von einem Bit; stellt einen mit einem Bit codierten Statuswert (der nur gelesen werden darf) dar, wobei der Wert „1“ den Status „ON“ und der Wert „0“ den Status „OFF“ codiert; auch und insbesondere im Zusammenhang mit Feldeingängen als „diskreter Eingang“ bezeichnet

### 3.1.6.3

#### **Discrete Inputs, diskrete Eingänge**

(en: discrete inputs, discretess)

Anwendungsprozessobjekt; eine Menge von Discrete Inputs, gekennzeichnet durch die Adresse eines Discrete Input und eine Menge von Discrete Inputs; diese Menge wird auch und insbesondere im Zusammenhang mit Feldeingängen als „diskrete Eingänge“ bezeichnet

### 3.1.6.4

#### **Coil, diskreter Ausgang**

(en: coil, discrete output)

Anwendungsprozessobjekt; adressiert durch eine vorzeichenlose Zahl und mit einer Breite von einem Bit; stellt einen mit einem Bit codierten Statuswert (der gelesen und geschrieben werden darf) dar, wobei der Wert „1“ den Status „ON“ und der Wert „0“ den Status „OFF“ codiert; im Zusammenhang mit Feldausgängen auch als „diskreter Ausgang“ bezeichnet

### 3.1.6.5

#### **gekapselte Schnittstelle**

(en: encapsulated interface)

Mechanismus, bei dem ein Dienst für eine Schnittstelle gekapselt wird, wobei dieser einem Anwendungsprozessobjekt entspricht, das durch einen MEI-Typ gekennzeichnet ist

### 3.1.6.6

#### **Ausnahme**

(en: exception)

Codierung, die zur Signalisierung eines Fehlers der Dienstanforderung verwendet wird

### 3.1.6.7

#### **Ausnahmecode**

(en: exception code)

Codierung in Bezug auf eine Ausnahme, die Einzelheiten zum Grund eines Fehlers der Dienstanforderung angibt

### 3.1.6.8

#### **Datei**

(en: file)

Anwendungsprozessobjekt; Gliederung von Einträgen, gekennzeichnet durch eine vorzeichenlose Zahl

### 3.1.6.9

#### **Funktionscode**

(en: function code)

Codierung eines Dienstes, der von einem Server angefordert wird

### 3.1.6.10

#### **Holding Register, Ausgaberegister**

(en: holding register, output register)

Anwendungsprozessobjekt; adressiert durch eine vorzeichenlose Zahl; stellt Werte mit 16 Bits dar; auch und insbesondere im Zusammenhang mit Feldausgängen als „Ausgaberegister“ bezeichnet

### 3.1.6.11

#### **Holding Registers, Ausgaberegister**

(en: holding registers, output registers)

Anwendungsprozessobjekt; eine Menge von Holding Registers, gekennzeichnet durch die Adresse eines Holding Register und eine Menge von Holding Registers; auch und insbesondere im Zusammenhang mit Feldausgängen als „Ausgaberegister“ bezeichnet

### 3.1.6.12

#### **Input Register**

(en: input register)

Anwendungsprozessobjekt; adressiert durch eine vorzeichenlose Zahl; stellt Werte mit 16 Bits dar; darf nur gelesen werden

**3.1.6.13****Input Registers****(en: input registers)**

Anwendungsprozessobjekt; eine Menge von Input Registers, gekennzeichnet durch die Adresse eines Input Register und eine Menge von Input Registers

**3.1.6.14****Satz****(en: record)**

Anwendungsprozessobjekt; eine Menge zusammenhängender Register eines vorgegebenen Typs, gekennzeichnet durch die Adresse des ersten Registers und durch die Anzahl der Register; im Kontext dieser Definition wurden die involvierten Register auch als „Referenzen“ bezeichnet

**3.1.6.15****Referenz****(en: reference)**

veraltete Benennung für Register

**3.1.6.16****Referenztyp****(en: reference type)**

veraltete Benennung für Registertyp

**3.1.6.17****Untercode****(en: sub-code)**

Spezialisierung eines Funktionscodes

**3.1.6.18****Einheit-ID****(en: unit ID)**

Kennung von logischen Geräten

**3.1.6.19****MEI-Typ****(en: MEI type)**

als Oktettwert angegebener Typ, der verwendet wird, um einen Dienst an die entsprechende Schnittstelle im Rahmen des Mechanismus der gekapselten Schnittstelle zu senden

**3.1.7 Spezielle Begriffe zum Veröffentlichen/Abonnieren****3.1.7.1****Netzwerkobjekt****(en: network object)**

Publish/Subscribe-Anwendung, Reader oder Writer

**3.1.7.2****GUID****(en: GUID)**

global eindeutige Kennung von Netzwerkobjekten

**3.1.7.3****zusammengesetzter Zustand****(en: composite state)**

Attribute einer Menge von Netzwerkobjekten

**3.1.7.4****Reader****(en: reader)**

Subscriber oder ein CST-Reader

### 3.1.7.5

#### **Writer**

(en: **writer**)

Publisher oder ein CST-Writer

### 3.1.7.6

#### **CST-Reader**

(en: **CSTReader**)

auf Metainformationen spezialisierter Subscriber

### 3.1.7.7

#### **CST-Writer**

(en: **CSTWriter**)

auf Metainformationen spezialisierter Publisher

### 3.1.7.8

#### **Kommunikationsakteur**

(en: **communication actor**)

Reader oder Writer

### 3.1.7.9

#### **Domänenteilnehmer**

(en: **domain participant**)

Anwendung, die Publish/Subscribe-Elemente enthält; auch als Publish/Subscribe-Anwendung bezeichnet

**ANMERKUNG** Diese Terminologie wurde eingeführt, um den übermäßigen Gebrauch der Benennung „Anwendung“ zu vermeiden. Gleichzeitig nimmt die Benennung „Domäne“ einen Platz innerhalb von „Publish/Subscribe“ ein. Die Erweiterungsmöglichkeit des Typs lässt das Konzept der „Domänen“ oder unabhängigen Kommunikationsebenen zu, wodurch eine effektive Isolierung des Anwendungsaustausches innerhalb von Domänen zugelassen wird. Während der DDS von OMG, wie in „Data Distribution Service for Real-Time Systems Specification, Version 1.1, December 2005“, diese Erweiterung verwendet, wird diese Möglichkeit in der vorliegenden Spezifikation nicht weiter untersucht, sondern eine einzelne Domäne betrachtet.

### 3.1.7.10

#### **Manager**

(en: **manager**)

spezialisierte Publish/Subscribe-Anwendung, die spezialisierte Publisher und Subscriber umfasst und in den beschriebenen Erkennungs- und Wartungsmechanismus eingeschlossen ist; nicht zu verwechseln mit einem veröffentlichenden Manager

### 3.1.7.11

#### **verwalteter Teilnehmer**

(en: **managed participant**)

eine Publish/Subscribe-Anwendung; die Bezeichnung bezieht sich auf ihre Rolle in Bezug auf einen Manager innerhalb des beschriebenen Erkennungs- und Wartungsmechanismus; nicht zu verwechseln mit einem untergeordneten veröffentlichenden Manager

### 3.1.7.12

#### **Sequenznummer**

(en: **sequence number**)

Nummer zur eindeutigen Identifikation elementarer Publish/Subscribe-Nachrichten in ihrer Reihenfolge

**Industrial communication networks -  
Fieldbus specifications -  
Part 6-15: Application layer protocol specification -  
Type 15 elements  
(IEC 61158-6-15:2010)**

Réseaux de communication industriels -  
Spécifications des bus de terrain -  
Partie 6-15: Spécification des protocoles  
des couches d'application -  
Éléments de type 15  
(CEI 61158-6-15:2010)

Industrielle Kommunikationsnetze -  
Feldbusse -  
Teil 6-15: Protokollspezifikation des  
Application Layer (Anwendungsschicht) -  
Typ 15-Elemente  
(IEC 61158-6-15:2010)

This European Standard was approved by CENELEC on 2012-03-28. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

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

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

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

**CENELEC**

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

**Management Centre: Avenue Marnix 17, B - 1000 Brussels**

## Foreword

The text of document 65C/607/FDIS, future edition 2 of IEC 61158-6-15, prepared by SC 65C, "Industrial networks", of IEC/TC 65, "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61158-6-15:2012.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2012-12-28
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2015-03-28

This document supersedes EN 61158-6-15:2008.

EN 61158-6-15:2012 includes the following significant technical changes with respect to EN 61158-6-15:2008:

- editorial corrections.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

## Endorsement notice

The text of the International Standard IEC 61158-6-15:2010 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following note has to be added for the standard indicated:

IEC/TR 61158-1:2010      NOTE Harmonized as CLC/TR 61158-1:2010 (not modified).

## CONTENTS

|                                                                               |    |
|-------------------------------------------------------------------------------|----|
| INTRODUCTION .....                                                            | 7  |
| 1 Scope .....                                                                 | 8  |
| 1.1 General .....                                                             | 8  |
| 1.2 Specifications .....                                                      | 8  |
| 1.3 Conformance .....                                                         | 8  |
| 2 Normative references .....                                                  | 9  |
| 3 Terms and definitions, abbreviations, symbols and conventions .....         | 9  |
| 3.1 Terms and definitions .....                                               | 9  |
| 3.2 Abbreviations and symbols .....                                           | 16 |
| 3.3 Conventions .....                                                         | 17 |
| 3.4 Conventions used in state machines .....                                  | 19 |
| 4 Abstract syntax for client/server .....                                     | 20 |
| 5 Transfer syntax for client/server .....                                     | 21 |
| 5.1 General .....                                                             | 21 |
| 5.2 Common APDU structure .....                                               | 21 |
| 5.3 Service-specific APDU structures .....                                    | 24 |
| 5.4 Data representation 'on the wire' .....                                   | 46 |
| 6 Abstract syntax for publish/subscribe .....                                 | 46 |
| 7 Transfer syntax for publish/subscribe .....                                 | 46 |
| 7.1 General .....                                                             | 46 |
| 7.2 APDU structure .....                                                      | 47 |
| 7.3 Sub-message structure .....                                               | 47 |
| 7.4 APDU interpretation .....                                                 | 49 |
| 7.5 Service specific APDU structures .....                                    | 50 |
| 7.6 Common data representation for publish/subscribe .....                    | 70 |
| 8 Structure of FAL protocol state machines .....                              | 75 |
| 9 AP-context state machines for client/server .....                           | 77 |
| 10 FAL service protocol machine (FSPM) for client/server .....                | 77 |
| 10.1 General .....                                                            | 77 |
| 10.2 FSPM state tables .....                                                  | 77 |
| 10.3 Functions used by FSPM .....                                             | 84 |
| 10.4 Parameters of FSPM/ARPM primitives .....                                 | 84 |
| 10.5 Client/server server transactions .....                                  | 84 |
| 11 Application relationship protocol machines (ARPMs) for client/server ..... | 86 |
| 11.1 Application relationship protocol machines (ARPMs) .....                 | 86 |
| 11.2 AREP state machine primitive definitions .....                           | 87 |
| 11.3 AREP state machine functions .....                                       | 88 |
| 12 DLL mapping protocol machine (DMPM) for client/server .....                | 88 |
| 12.1 AREP mapping to data link layer .....                                    | 88 |
| 12.2 DMPM states .....                                                        | 89 |
| 12.3 DMPM state machine .....                                                 | 89 |
| 12.4 Primitives exchanged between data link layer and DMPM .....              | 90 |
| 12.5 Client/server on TCP/IP .....                                            | 90 |
| 13 AP-Context state machines for publish/subscribe .....                      | 94 |

|                                                                                                                                |    |
|--------------------------------------------------------------------------------------------------------------------------------|----|
| 14 Protocol machines for publish/subscribe .....                                                                               | 94 |
| 14.1 General.....                                                                                                              | 94 |
| 14.2 Publish/subscribe on UDP .....                                                                                            | 96 |
| Bibliography .....                                                                                                             | 97 |
| Annex ZA (normative) Normative references to international publications<br>with their corresponding European publications..... | 98 |
| Figure 1 – APDU Format .....                                                                                                   | 21 |
| Figure 2 – Client to server confirmed service request.....                                                                     | 22 |
| Figure 3 – Normal response from server to client .....                                                                         | 23 |
| Figure 4 – Exception response from server to client .....                                                                      | 23 |
| Figure 5 – Client to server unconfirmed service request.....                                                                   | 24 |
| Figure 6 – Publish/subscribe APDU .....                                                                                        | 46 |
| Figure 7 – Flags of issue request .....                                                                                        | 51 |
| Figure 8 – Flags of heartbeat request .....                                                                                    | 53 |
| Figure 9 – Flags of VAR request .....                                                                                          | 56 |
| Figure 10 – Flags of GAP request .....                                                                                         | 58 |
| Figure 11 – Flags of ACK request .....                                                                                         | 60 |
| Figure 12 – Flags of INFO_DST request.....                                                                                     | 63 |
| Figure 13 – Flags of INFO_REPLY request .....                                                                                  | 65 |
| Figure 14 – Flags of INFO_SRC request.....                                                                                     | 67 |
| Figure 15 – Flags of INFO_TS request.....                                                                                      | 68 |
| Figure 16 – Flags of PAD request .....                                                                                         | 70 |
| Figure 17 – Encoding of octet.....                                                                                             | 71 |
| Figure 18 – Encoding of boolean.....                                                                                           | 72 |
| Figure 19 – Encoding of unsigned short.....                                                                                    | 72 |
| Figure 20 – Encoding of unsigned long .....                                                                                    | 72 |
| Figure 21 – Encoding of unsigned long long .....                                                                               | 73 |
| Figure 22 – Encoding of float.....                                                                                             | 73 |
| Figure 23 – Encoding of double.....                                                                                            | 73 |
| Figure 24 – Relationships among protocol machines and adjacent layers .....                                                    | 76 |
| Figure 25 – State transition diagram of FSPM .....                                                                             | 77 |
| Figure 26 – Transaction state machine, per connection .....                                                                    | 78 |
| Figure 27 – Client/server server transactions.....                                                                             | 85 |
| Figure 28 – State transition diagram of the Client ARPM.....                                                                   | 86 |
| Figure 29 – State transition diagram of the server ARPM.....                                                                   | 87 |
| Figure 30 – State transition diagram of DMPM .....                                                                             | 89 |
| Figure 31 – APDU Format .....                                                                                                  | 90 |
| Figure 32 – TCP/IP PDU Format.....                                                                                             | 91 |
| Figure 33 – Publish/subscribe receiver .....                                                                                   | 95 |
| Table 1 – Conventions used for state machines .....                                                                            | 20 |
| Table 2 – Exception code .....                                                                                                 | 23 |
| Table 3 – Read discrettes request.....                                                                                         | 24 |

|                                                                                |    |
|--------------------------------------------------------------------------------|----|
| Table 4 – Read discretely response.....                                        | 25 |
| Table 5 – Read coils request.....                                              | 25 |
| Table 6 – Read coils response.....                                             | 26 |
| Table 7 – Write single coil request.....                                       | 26 |
| Table 8 – Write single coil response.....                                      | 27 |
| Table 9 – Write multiple coils request.....                                    | 27 |
| Table 10 – Write multiple coils response.....                                  | 28 |
| Table 11 – Broadcast write single coil request.....                            | 28 |
| Table 12 – Broadcast write multiple coils request.....                         | 29 |
| Table 13 – Read input registers request.....                                   | 30 |
| Table 14 – Read input registers response.....                                  | 30 |
| Table 15 – Read holding registers request.....                                 | 31 |
| Table 16 – Read holding registers response.....                                | 31 |
| Table 17 – Write single holding register request.....                          | 32 |
| Table 18 – Write single holding register response.....                         | 32 |
| Table 19 – Write multiple holding registers request.....                       | 33 |
| Table 20 – Write multiple holding registers response.....                      | 33 |
| Table 21 – Mask write holding register request.....                            | 34 |
| Table 22 – Mask write holding register request.....                            | 34 |
| Table 23 – Read/Write multiple holding registers request.....                  | 35 |
| Table 24 – Read/Write multiple holding registers response.....                 | 35 |
| Table 25 – Read FIFO request.....                                              | 36 |
| Table 26 – Read FIFO response.....                                             | 36 |
| Table 27 – Broadcast write single holding register request.....                | 37 |
| Table 28 – Broadcast write multiple holding registers request.....             | 38 |
| Table 29 – Read file record request.....                                       | 38 |
| Table 30 – Read file record response.....                                      | 39 |
| Table 31 – Write file record request.....                                      | 40 |
| Table 32 – Write file record response.....                                     | 42 |
| Table 33 – Read device identification request.....                             | 43 |
| Table 34 – Device identification categories.....                               | 43 |
| Table 35 – Read device ID code.....                                            | 44 |
| Table 36 – Read device identification response.....                            | 44 |
| Table 37 – Conformity level.....                                               | 45 |
| Table 38 – Requested vs. returned known objects.....                           | 45 |
| Table 39 – APDU structure.....                                                 | 47 |
| Table 40 – Sub-message structure.....                                          | 48 |
| Table 41 – Publish/subscribe service identifier encoding.....                  | 48 |
| Table 42 – Attributes changed modally and affecting APDUs interpretations..... | 49 |
| Table 43 – Issue request.....                                                  | 51 |
| Table 44 – Meaning of issue request flags.....                                 | 52 |
| Table 45 – Interpretation of issue.....                                        | 52 |
| Table 46 – Heartbeat request.....                                              | 53 |

|                                                                                 |    |
|---------------------------------------------------------------------------------|----|
| Table 47 – Meaning of heartbeat request flags .....                             | 54 |
| Table 48 – Interpretation of heartbeat .....                                    | 54 |
| Table 49 – VAR request .....                                                    | 55 |
| Table 50 – Meaning of VAR request flags.....                                    | 56 |
| Table 51 – Interpretation of VAR .....                                          | 57 |
| Table 52 – GAP request .....                                                    | 58 |
| Table 53 – Meaning of GAP request flags.....                                    | 59 |
| Table 54 – Interpretation of GAP .....                                          | 59 |
| Table 55 – ACK request .....                                                    | 60 |
| Table 56 – Meaning of ACK request flags.....                                    | 61 |
| Table 57 – Interpretation of ACK .....                                          | 61 |
| Table 58 – Header request .....                                                 | 62 |
| Table 59 – Change in state of the receiver.....                                 | 62 |
| Table 60 – INFO_DST request.....                                                | 63 |
| Table 61 – Meaning of INFO_DST request flags .....                              | 64 |
| Table 62 – INFO_REPLY request .....                                             | 65 |
| Table 63 – Meaning of INFO_REPLY request flags.....                             | 65 |
| Table 64 – INFO_SRC request.....                                                | 66 |
| Table 65 – Meaning of INFO_SRC request flags .....                              | 67 |
| Table 66 – INFO_TS request.....                                                 | 68 |
| Table 67 – Meaning of INFO_TS request flags .....                               | 69 |
| Table 68 – PAD request .....                                                    | 69 |
| Table 69 – Meaning of PAD request flags.....                                    | 70 |
| Table 70 – Semantics .....                                                      | 71 |
| Table 71 – FSPM state table – client transactions .....                         | 79 |
| Table 72 – FSPM state table – server transactions.....                          | 84 |
| Table 73 – Function MatchInvokeID().....                                        | 84 |
| Table 74 – Function HighBit().....                                              | 84 |
| Table 75 – Parameters used with primitives exchanged between FSPM and ARPM..... | 84 |
| Table 76 – Client ARPM states.....                                              | 86 |
| Table 77 – Client ARPM state table.....                                         | 86 |
| Table 78 – Server ARPM states .....                                             | 86 |
| Table 79 – Server ARPM state table .....                                        | 87 |
| Table 80 – Primitives issued from ARPM to DMPM .....                            | 87 |
| Table 81 – Primitives issued by DMPM to ARPM .....                              | 87 |
| Table 82 – Parameters used with primitives exchanged between ARPM and DMPM..... | 88 |
| Table 83 – DMPM state descriptions.....                                         | 89 |
| Table 84 – DMPM state table – client transactions .....                         | 89 |
| Table 85 – DMPM state table – server transactions.....                          | 90 |
| Table 86 – Primitives exchanged between data-link layer and DMPM .....          | 90 |
| Table 87 – Encapsulation parameters for client/server on TCP/IP .....           | 91 |

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

This 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.

## INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

### Part 6-15: Application layer protocol specification – Type 15 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 15 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible behavior provided by the Type 15 fieldbus Application Layer in terms of

- a) the abstract syntax defining the application layer protocol data units conveyed between communicating application entities,
- b) the transfer syntax defining the application layer protocol data units conveyed between communicating application entities,
- c) the application context state machine defining the application service behavior visible between communicating application entities; and
- d) the application relationship state machines defining the communication behavior visible between communicating application entities; and.

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

- a) define the wire-representation of the service primitives defined in IEC 61158-5-15, and
- b) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 15 IEC 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 61158-5-15.

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. 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 61158-5-15:2010<sup>1</sup>, *Industrial communication networks – Fieldbus specifications - Part 5-15: Application layer service definition – Type 15 elements*

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

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

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

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

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

### 3.1 Terms and definitions

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

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

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

#### 3.1.2 ISO/IEC 8822 terms

- a) abstract syntax
- b) presentation context

#### 3.1.3 ISO/IEC 9545 terms

- a) application-association
- b) application-context
- c) application context name
- d) application-entity-invocation
- e) application-entity-type
- f) application-process-invocation

---

<sup>1</sup> To be published.

- g) application-process-type
- h) application-service-element
- i) application control service element

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

- a) object identifier
- b) type

#### **3.1.5 IEC/TR 61158-1 terms**

The following IEC/TR 61158-1 terms apply.

##### **3.1.5.1**

##### **application**

function or data structure for which data is consumed or produced

##### **3.1.5.2**

##### **application layer interoperability**

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

##### **3.1.5.3**

##### **application object**

object class that manages and provides the run time exchange of messages across the network and within the network device

NOTE Multiple types of application object classes may be defined.

##### **3.1.5.4**

##### **application process**

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

##### **3.1.5.5**

##### **application process identifier**

distinguishes multiple application processes used in a device

##### **3.1.5.6**

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

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

NOTE Application process object definitions are composed of a set of values for the attributes of their class.

##### **3.1.5.7**

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

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

##### **3.1.5.8**

##### **application relationship**

cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

NOTE This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities.

##### **3.1.5.9**

##### **application relationship endpoint**

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

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

#### **3.1.5.10**

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

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

#### **3.1.5.11**

##### **attribute**

description of an externally visible characteristic or feature of an object

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

#### **3.1.5.12**

##### **behavior**

indication of how the object responds to particular events

NOTE Its description includes the relationship between attribute values and services.

#### **3.1.5.13**

##### **class**

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

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

#### **3.1.5.14**

##### **class attributes**

attribute that is shared by all objects within the same class

#### **3.1.5.15**

##### **class code**

unique identifier assigned to each object class

#### **3.1.5.16**

##### **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.

#### **3.1.5.17**

##### **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, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

#### **3.1.5.18**

##### **conveyance path**

unidirectional flow of APDUs across an application relationship

#### **3.1.5.19**

##### **cyclic**

term used to describe events which repeat in a regular and repetitive manner

#### **3.1.5.20**

##### **dedicated AR**

AR used directly by the FAL user

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

#### 3.1.5.21

##### **device**

physical hardware connection to the link

NOTE A device may contain more than one node.

#### 3.1.5.22

##### **device profile**

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

#### 3.1.5.23

##### **dynamic AR**

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

#### 3.1.5.24

##### **endpoint**

one of the communicating entities involved in a connection

#### 3.1.5.25

##### **error**

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

#### 3.1.5.26

##### **error class**

general grouping for error definitions

NOTE Error codes for specific errors are defined within an error class.

#### 3.1.5.27

##### **error code**

identification of a specific type of error within an error class

#### 3.1.5.28

##### **FAL subnet**

networks composed of one or more data link segments

NOTE Subnets are permitted to contain bridges, but not routers. FAL subnets are identified by a subset of the network address.

#### 3.1.5.29

##### **logical device**

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

#### 3.1.5.30

##### **management information**

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

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

#### 3.1.5.31

##### **network**

series of nodes connected by some type of communication medium

NOTE The connection paths between any pair of nodes can include repeaters, routers and gateways.

### 3.1.5.32

#### **peer**

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

### 3.1.5.33

#### **pre-defined AR endpoint**

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

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

### 3.1.5.34

#### **pre-established AR endpoint**

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

### 3.1.5.35

#### **Publisher**

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

NOTE The publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated AR. Two types of publishers are defined by this standard, Pull Publishers and Push Publishers, each of which is defined separately.

### 3.1.5.36

#### **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

### 3.1.5.37

#### **service**

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

NOTE A set of common services is defined and provisions for the definition of object-specific services are provided. Object-specific services are those which are defined by a particular object class to perform a required function which is not performed by a common service.

### 3.1.5.38

#### **subscriber**

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

NOTE Two types of subscribers are defined by this standard, pull subscribers and push subscribers, each of which is defined separately.

## 3.1.6 Specific definitions for client/server

### 3.1.6.1

#### **coils, discrete outputs**

application process object, a set of coils, characterized by the address of a coil and a quantity of coils, this set is also called discrete outputs when associated with field outputs

### 3.1.6.2

#### **discrete, discrete input**

application process object, addressed by an unsigned number and having a width of one bit, representing a 1-bit encoded status value, read-only, with the value '1' encoding the status ON and the value '0' encoding the status OFF, also called discrete input, especially when associated with field inputs

### 3.1.6.3

#### **discrete inputs, discretes**

application process object, a set of discretes, characterized by the address of a discrete and a quantity of discretes, this set is also called discrete inputs, especially when associated with field inputs

#### 3.1.6.4

##### **coil, discrete output**

application process object, addressed by an unsigned number and having a width of one bit, representing a 1-bit encoded status value, read-write, with the value '1' encoding the status ON and the value '0' encoding the status OFF, also called discrete output when associated with field output

#### 3.1.6.5

##### **encapsulated interface**

mechanism encapsulating a service for an interface, which is an application process object characterized by an MEI type

#### 3.1.6.6

##### **exception**

encoding used to signal a service request failure

#### 3.1.6.7

##### **exception code**

encoding associated with an exception, detailing the reason of a service request failure

#### 3.1.6.8

##### **file**

application process object, an organization of records, characterized by an unsigned number

#### 3.1.6.9

##### **function code**

encoding of a service requested to a server

#### 3.1.6.10

##### **holding register, output register**

application process object, addressed by an unsigned number and representing values with 16 bits, read-write, also called output register, especially when associated with field outputs

#### 3.1.6.11

##### **holding registers, output registers**

application process object, a set of holding registers, characterized by the address of a holding register and a quantity of holding registers, also called output registers, especially when associated with field outputs

#### 3.1.6.12

##### **input register**

application process object, addressed by an unsigned number and representing values with 16 bits, read-only

#### 3.1.6.13

##### **input registers**

application process object, a set of input registers, characterized by the address of an input register and a quantity of input registers

#### 3.1.6.14

##### **record**

application process object, a set of contiguous registers of a specified type, characterized by the address of the first register and by the quantity of registers; in the context of this definition, the registers involved have also been called references

#### 3.1.6.15

##### **reference**

denigrated term for register

#### **3.1.6.16**

##### **reference type**

denigrated term for register type

#### **3.1.6.17**

##### **sub-code**

specialization of a function code

#### **3.1.6.18**

##### **unit ID**

logical device identifier

#### **3.1.6.19**

##### **MEI type**

type specified as an octet value, used to dispatch a service to the appropriate interface in the context of the encapsulated interface mechanism

### **3.1.7 Specific definitions for publish/subscribe**

#### **3.1.7.1**

##### **network object**

Publish/subscribe application, reader, or writer

#### **3.1.7.2**

##### **GUID**

globally unique network object identifier

#### **3.1.7.3**

##### **composite state**

attributes of a set of network objects

#### **3.1.7.4**

##### **reader**

subscriber or a CSTReader

#### **3.1.7.5**

##### **writer**

Publisher or a CSTWriter

#### **3.1.7.6**

##### **CSTReader**

meta-information-specialized subscriber

#### **3.1.7.7**

##### **CSTWriter**

meta-information-specialized publisher

#### **3.1.7.8**

##### **communication actor**

reader or writer

#### **3.1.7.9**

##### **domain participant**

application that contains publish/subscribe elements, also called publish/subscribe application

NOTE This terminology is adopted to avoid the overuse of the term "application". At the same time, the term "domain" has a place within publish/subscribe. The Type extensibility allows for the concept of "domains", or independent communication planes, effectively permitting isolation of application exchanges within domains. While OMG DDS as in "Data Distribution Service for Real-Time Systems

Specification, Version 1.1, December 2005" uses this extension, the feature will not be examined further in this specification, which will consider a single domain.

#### **3.1.7.10**

##### **manager**

specialized publish/subscribe application, containing specialized publishers and subscribers, and involved in the described discovery and maintenance mechanism; not to be confused with any publishing manager

#### **3.1.7.11**

##### **managed participant**

a publish/subscribe application; the qualifier refers to its role in relation to a manager when involved in the described discovery and maintenance mechanism; not to be confused with any publishing manager subordinate

#### **3.1.7.12**

##### **sequence number**

number used to uniquely identify elementary publish/subscribe messages in an ordered manner

### **3.2 Abbreviations and symbols**

#### **3.2.1 Common abbreviations and symbols**

|       |                                                    |
|-------|----------------------------------------------------|
| AE    | Application Entity                                 |
| AL    | Application Layer                                  |
| ALME  | Application Layer Management Entity                |
| ALP   | Application Layer Protocol                         |
| APO   | Application Object                                 |
| AP    | Application Process                                |
| APDU  | Application Protocol Data Unit                     |
| API   | Application Process Identifier                     |
| AR    | Application Relationship                           |
| AREP  | Application Relationship End Point                 |
| ASCII | American Standard Code for Information Interchange |
| ASE   | Application Service Element                        |
| Cnf   | Confirmation                                       |
| DL-   | (as a prefix) data-link-                           |
| DLC   | data-link Connection                               |
| DLCEP | data-link Connection End Point                     |
| DLL   | data-link Layer                                    |
| DLM   | data-link-management                               |
| DLSAP | data-link Service Access Point                     |
| DLSDU | DL-service-data-unit                               |
| FAL   | Fieldbus Application Layer                         |
| ID    | Identifier                                         |
| IEC   | International Electrotechnical Commission          |
| Ind   | Indication                                         |
| LME   | Layer Management Entity                            |
| lsb   | Least Significant Bit                              |
| msb   | Most Significant Bit                               |

|      |                                    |
|------|------------------------------------|
| OSI  | Open Systems Interconnect          |
| QoS  | Quality of Service                 |
| Req  | Request                            |
| Rsp  | Response                           |
| SAP  | Service Access Point               |
| SDU  | Service Data Unit                  |
| SMIB | System Management Information Base |
| SMK  | System Management Kernel           |

### 3.2.2 Abbreviations and symbols for client/server

|     |                             |
|-----|-----------------------------|
| C/S | Client/Server               |
| FC  | Function Code               |
| CAN | Controller Area Network     |
| CiA | CAN in Automation           |
| MEI | Encapsulated Interface type |
| URL | Uniform Resource Locator    |

### 3.2.3 Abbreviations and symbols for publish/subscribe

|      |                                 |
|------|---------------------------------|
| CS   | Composite State                 |
| DCPS | Data-Centric Publish-Subscribe  |
| DDS  | Data Distribution Service       |
| DLRL | Data Local Reconstruction Layer |
| OMG  | Object Management Group         |
| P/S  | Publish/Subscribe               |

## 3.3 Conventions

### 3.3.1 Overview

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

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

### 3.3.2 General conventions

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

### 3.3.3 Conventions for class definitions

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

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

- (1) The "FAL ASE:" entry is the name of the FAL ASE that provides the services for the class being specified.
- (2) The "CLASS:" entry is the name of the class being specified. All objects defined using this template will be an instance of this class. The class may be specified by this standard, or by a user of this standard.
- (3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the FAL ASE that will provide the services for this class. When qualified by the identity of its FAL ASE, it unambiguously identifies the class within the scope of the FAL. The value "NULL" indicates that the class cannot be instantiated. Class IDs between 1 and 255 are reserved by this standard to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2 048 are allocated for identifying user defined classes.
- (4) The "PARENT CLASS:" entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited for the class being defined, and therefore do not have to be redefined in the template for this class.

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

- (5) The "ATTRIBUTES" label indicate that the following entries are attributes defined for the class.
  - a) Each of the attribute entries contains a line number in column 1, a mandatory (m) / optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label in column 3, a name or a conditional expression in column 4, and optionally a list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.
  - b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.
  - c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting is used to specify

- i) fields of a structured attribute (4.1, 4.2, 4.3),
- ii) attributes conditional on a constraint statement (5). Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2).
- iii) the selection fields of a choice type attribute (6.1 and 6.2).

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

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

### 3.3.4 Conventions for service definitions

#### 3.3.4.1 General

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 61158-5-15. 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 61158-5-15. The service specification defines the services that are provided by the ASE.

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

### 3.3.5 Conventions for class definitions

The data-link layer mapping definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is defined in IEC/TR 61158-1.

### 3.3.6 Abstract syntax conventions

When the "optionalParametersMap" parameter is used, a bit number which corresponds to each OPTIONAL or DEFAULT production is given as a comment.

## 3.4 Conventions used in state machines

The state machines are described in Table 1:

**Table 1 – Conventions used for state machines**

| #                        | Current state                                             | Event / condition<br>=> action                                                                                                                                                                         | Next state                                                    |
|--------------------------|-----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Name of this transition. | The current state to which this state transition applies. | Events or conditions that trigger this state transaction.<br>=><br>The actions that are taken when the above events or conditions are met. The actions are always indented below events or conditions. | The next state after the actions in this transition is taken. |

The conventions used in the state machines are as follows:

**:=** Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.

**xxx** A parameter name.

**EXAMPLE 1**

Identifier := reason

means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'

"xxx" Indicates fixed value.

**EXAMPLE 2**

Identifier := "abc"

means value "abc" is assigned to a parameter named 'Identifier.'

- = A logical condition to indicate an item on the left is equal to an item on the right.
- < A logical condition to indicate an item on the left is less than the item on the right.
- > A logical condition to indicate an item on the left is greater than the item on the right.
- <> A logical condition to indicate an item on the left is not equal to an item on the right.
- && Logical "AND"
- || Logical "OR"

This construct allows the execution of a sequence of actions in a loop within one transition. The loop is executed for all values from start\_value to end\_value.

**EXAMPLE 3**

```
for (Identifier := start_value to end_value)
  actions
endfor
```

This construct allows the execution of alternative actions depending on some condition (which might be the value of some identifier or the outcome of a previous action) within one transition.

**EXAMPLE 4**

```
If (condition)
  actions
else
  actions
endif
```

Readers are strongly recommended to refer to the subclauses for the AREP attribute definitions, the local functions, and the FAL-PDU definitions to understand protocol machines. It is assumed that readers have sufficient knowledge of these definitions, and they are used without further explanations.

## 4 Abstract syntax for client/server

The abstract syntax of APDUs is combined with their transfer syntax and is specified in Clause 5.

## 5 Transfer syntax for client/server

### 5.1 General

The sending Application Layer prepares an APDU to transfer to the receiving Application Layer. It uses the parameters from the service primitives to do so. There are several formats of the APDU:

- request APDU from Client to Server device or devices, or
- normal response and positive confirmation from Server to Client device, or
- response and negative confirmation from Server to Client device.

The format and coding rules for all APDUs are specified in this clause.

### 5.2 Common APDU structure

All APDUs have a common structure as shown in Figure 1.

| Unit ID | Code | Data |
|---------|------|------|
|---------|------|------|

Figure 1 – APDU Format

#### 5.2.1 Unit ID

Client/server devices in the role of servers are addressed using a Unit ID. The Unit ID needs to be unique across all the servers addressable by a client. The set of addressable servers is determined by the underlying layer. This set is sometimes called connection.

The Unit ID assignment is outside of the scope of this specification.

The Unit ID identifies logical devices. There may be more than one logical device per physical device.

Some values of Unit ID are reserved and have particular meanings. The value 0 is reserved for broadcast.

Field Type: Unsigned8

Allowed values: 1 to 247, and 0 for broadcast where supported.

**NOTE** In general the Unit ID is only required for logical devices having the role of servers. Often logical devices can have either role or multiple roles, via configuration, and their Unit ID is not used when they only perform in the client role. Depending on the underlying layers some devices can have concurrently a client and a server role on the same access point, this is the case for client/server on Token Bus/HDLC, outside the scope of this specification.

#### 5.2.2 Code

##### 5.2.2.1 General

This field represents:

- a requested service, confirmed or unconfirmed, via an identifier called function code, or
- the normal response and positive confirmation of a requested service, represented by echoing the requested service identifier, or
- the exception response and negative confirmation of a requested service, represented by echoing the requested service identifier with its high bit turned on. The latter representation is also called exception.

Field Type: Unsigned8

### 5.2.2.2 Service identifiers and function codes

Client/server service identifiers are commonly called function codes.

Function codes are encodings of services requested to a server. Some function codes are further specialized by means of a sub-code, specified as part of the data field. These encodings are partitioned in three categories, and since the subdivision may propagate to the sub-codes, for sake of completeness, despite being part of the data field they will also be mentioned here:

#### Publicly assigned function codes

These function codes are either assigned to a standard service or reserved for future assignment. The standard services and their identifiers will be detailed in this specification.

#### User definable function codes

These function codes can be used for experimentation in a controlled laboratory environment. They must not be used in an open environment.

Ranges: There are two ranges, FC 65 (0x41) to 72 (0x48) included, and 100 (0x64) to 110 (0x6E) included.

#### Reserved function codes

These function codes are currently used by some companies for legacy products and are not available for public use.

NOTE 1 Function code assignments are managed by the Modbus-IDA industrial consortium.

NOTE 2 The following function codes, while publicly assigned, are not covered by this specification: FC 7 (0x07, Read Exception Status), FC 8 (0x08, Diagnostics), FC 11 (0x0B, Get Com Event Counter), FC 12 (0x0C, Get Com Event Log), FC 17 (0x11, Report Slave ID).

NOTE 3 The following function codes and function code/sub-codes are reserved: FC 8/19 (0x08/0x13), FC 8/21-255 (0x08/0x15-0xFFFF), FC 9 (0x09), FC 10 (0x0A), FC 13 (0x0D), FC 14 (0x0E), FC 41 (0x29), FC 42 (0x2A), FC 43/0-12 (0x2B/0x00-0x0C), FC 43/15-255 (0x2B/0x0F-0xFF), FC 90 (0x5A), FC 91 (0x5B), FC 125 (0x7D), FC 126 (0x7E), FC 127 (0x7F).

### 5.2.3 Data

For normal requests and responses this is the user data which is transferred between the application layer and its user. The application layer assembles it from the parameters of a service primitive or parses it into parameters of a service primitive. Its structure depends upon the type of APDU. For exception responses it represents the reason for the exception via an exception code.

Field Type: from 1 to 252 octets; the Type is APDU specific.

### 5.2.4 Client to server confirmed service request

The format is as in Figure 2.

|         |               |              |
|---------|---------------|--------------|
| Unit ID | Function code | Request Data |
|---------|---------------|--------------|

Figure 2 – Client to server confirmed service request

### 5.2.5 Normal response from server to client

The format for the normal response to a confirmed service is as in Figure 3. The Unit ID and the function code fields are the same as the corresponding fields in the request.

|         |               |               |
|---------|---------------|---------------|
| Unit ID | Function code | Response Data |
|---------|---------------|---------------|

**Figure 3 – Normal response from server to client****5.2.6 Exception response from server to client**

The format for the exception response is as in Figure 4. The Unit ID is the same as the corresponding field in the request. The exception is produced adding 0x80 to the function code of the corresponding request.

|         |           |                |
|---------|-----------|----------------|
| Unit ID | Exception | Exception code |
|---------|-----------|----------------|

**Figure 4 – Exception response from server to client**

The exception codes, giving information on the service failure, are shown in Table 2.

**Table 2 – Exception code**

| Encoding | Name                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01     | Illegal function         | The function code received in the query is not an allowable action for the server. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server is in the wrong state to process a request of this type, for example because it is not configured and it is being asked to return register values                                                                                            |
| 0x02     | Illegal data address     | The data address received in the query is not an allowable address for the server. More specifically, the combination of reference number and transfer length is invalid. Example For a controller with 100 registers, a request with offset (data address) 96 and length 4 would succeed, a request with offset 96 and length 5 would generate exception code 0x02                                                                                                                             |
| 0x03     | Illegal data value       | A value contained in the query data field is not an allowable value for server. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean for example that a data item submitted for storage in a register has a value outside the expectation of the application program, since the client/server protocol is unaware of the significance of any particular value for any particular register |
| 0x04     | Server device failure    | An unrecoverable error occurred while the server was attempting to perform the requested action                                                                                                                                                                                                                                                                                                                                                                                                 |
| 0x05     | Acknowledge              | The server accepted the service invocation but the service requires a relatively long time to execute. The server therefore returns only an acknowledgement of the service invocation receipt. This response is returned to prevent a timeout error from occurring in the client                                                                                                                                                                                                                |
| 0x06     | Server busy              | The server was unable to accept the request. The client application has the responsibility of deciding if and when to re-send the request                                                                                                                                                                                                                                                                                                                                                       |
| 0x08     | Memory parity error      | For specialized use in conjunction with function codes 20 (0x14) and 21 (0x15), to indicate that the extended file area failed to pass a consistency check. For example, the server attempted to read record file, but detected a memory parity error. The client can retry the request, but service may be required on the server device                                                                                                                                                       |
| 0x0A     | Gateway path unavailable | For specialized use in conjunction with gateways, hubs, switches and similar network devices, to indicate that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. This usually means that the gateway is misconfigured or overloaded                                                                                                                                                                          |