

DIN EN 61158-6-2

ICS 35.100. 70; 35.160

Ersatz für
DIN EN 61158-6-2:2013-03
Siehe Anwendungsbeginn

**Industrielle Kommunikationsnetze –
Feldbusse –
Teil 6-2: Protokollspezifikation des Application Layer
(Anwendungsschicht) – Typ 2-Elemente
(IEC 61158-6-2:2014);
Englische Fassung EN 61158-6-2:2014, nur auf CD-ROM**

Industrial communication networks –
Fieldbus specifications –
Part 6-2: Application layer protocol specification – Type 2 elements
(IEC 61158-6-2:2014);
English version EN 61158-6-2:2014, only on CD-ROM

Réseaux de communication industriels –
Spécifications des bus de terrain –
Partie 6-2: Spécification du protocole de la couche application – Éléments de type 2
(CEI 61158-6-2:2014);
Version anglaise EN 61158-6-2:2014, seulement en CD-ROM

Gesamtumfang 271 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Anwendungsbeginn

Anwendungsbeginn für die von CENELEC am 2014-09-23 angenommene Europäische Norm als DIN-Norm ist 2015-09-01.

Für DIN EN 61158-6-2:2013-03 besteht eine Übergangsfrist bis 2017-09-23.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN EN 61158-600:2013-08.

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

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

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (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 ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnitsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

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

Der Zusammenhang der zitierten Dokumente mit den entsprechenden Deutschen Dokumenten 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 61158-6-2 veröffentlicht. Zu deren Abschnitt 1, der den Anwendungsbereich festlegt, und Abschnitt 3, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als 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 2 fest.

Änderungen

Gegenüber DIN EN 61158-6-2:2013-03 wurden folgende Änderungen vorgenommen:

- a) „Member“ und objektspezifische Dienste in 4.1.2.1, 4.1.8, 4.1.10 und 8.2 hinzugefügt;
- b) Entfernung von überholten Transportklassen 4 bis 6 in 4.1.4.6 und 9.3.9 bis 9.3.12;
- c) Definitionen, verwendet von Time Sync-Objekt, auf den neuesten Stand gebracht;
- d) Klarstellung der Transport-Dateikopfformate in 4.1.4;
- e) CM- und MR-PDUs in 4.1.5 bis 4.1.7 auf den neuesten Stand gebracht;
- f) Identity Objekt-PDUs in 4.1.8.2 auf den neuesten Stand gebracht;
- g) Assembly Objekt-PDUs in 4.1.8.4 auf den neuesten Stand gebracht;
- h) Time Sync-Objekt-PDUs in 4.1.8.6 auf den neuesten Stand gebracht;
- i) Parameter Objekt-PDUs in 4.1.8.7 auf den neuesten Stand gebracht;
- j) Connection Manager-Objekt-PDU in 4.1.8.8 auf den neuesten Stand gebracht;
- k) Nachrichten- und Verbindungswege in 4.1.9 auf den neuesten Stand gebracht;
- l) Fehlercodes in 4.1.11 und Objektklassencode in 4.1.10 auf den neuesten Stand gebracht;
- m) Encapsulation abstract syntax in 4.3 auf den neuesten Stand gebracht;
- n) DLL Mapping Protokollmaschine 2 in Abschnitt 11 auf den neuesten Stand gebracht;
- o) redaktionelle Überarbeitung;
- p) Datentypen in 4.2.4 und 5.2.3 auf den neuesten Stand gebracht.

Frühere Ausgaben

DIN V 19245-1: 1988-01

DIN EN 50170/A3: 2002-11

DIN EN 61158-6: 2005-05

DIN EN 61158-6-2: 2008-09, 2013-03

Nationaler Anhang NA (informativ)

1 Anwendungsbereich^{N1)}

1.1 Allgemeines

Der Feldbus-Application Layer (FAL) liefert Benutzerprogramme mit Hilfsmitteln für den Zugriff auf die Feldbuskommunikationsumgebung. In dieser Hinsicht kann der FAL als ein Fenster zwischen zugehörigen Anwendungsprogrammen betrachtet werden.

Diese Norm stellt für Typ 2-Feldbusse maßgebliche allgemeine Elemente für zeitkritischen und nicht zeitkritischen Datentransfer zwischen Anwendungsprogrammen in einer Automatisierungsumgebung bereit. Der Begriff „zeitkritisch“ (en: time-critical) wird verwendet, um die Präsenz eines Zeitfensters darzustellen, in dem eine oder mehrere festgelegte Aktivitäten mit einem vorbestimmten Grad an Gewissheit erledigt werden. Das Scheitern der Erledigung der festgelegten Aktivität innerhalb des Zeitfensters trägt das Risiko des Betriebsausfalls der Anwendung mit und somit das aufgeförderte Eingreifen mit dazugehörigem Risiko für Geräte, Anlagen und möglicherweise Menschenleben.

Diese Norm legt Interaktionen zwischen entfernten Anwendungen fest und definiert das von außen sichtbare Verhalten, das vom Typ 2 des Feldbus-Application Layer bereitgestellt wird, in Form von:

- a) einer formellen abstrakten Syntax zur Festlegung der Application Layer-Protokoll-Dateneinheiten, die zwischen kommunizierenden Anwendungseinheiten übermittelt werden;
- b) der übertragenen Syntax zur Festlegung der Codierungsregeln, welche auf die Application Layer-Dateneinheiten angewendet sind;
- c) des Anwendungskontext der Zustandsmaschine zur Festlegung des sichtbaren Dienstverhaltens zwischen kommunizierenden Anwendungseinheiten;
- d) der Anwendungsbeziehung der Zustandsmaschine zur Festlegung des sichtbaren Kommunikationsverhaltens zwischen kommunizierenden Anwendungseinheiten.

Der Zweck dieser Norm ist, das Protokoll zu definieren, welches bereitgestellt wird zur

- Festlegung der Leitungsdarstellung der Dienstprimitiven, welche in IEC 61158-5-2 beschrieben sind, und
- zur Festlegung des von außen sichtbaren Verhaltens verbunden mit ihrer Übertragung.

Diese Norm beschreibt das Protokoll des Typ 2-Feldbus-Application Layer in Übereinstimmung mit dem OSI-Basis-Referenzmodell (ISO/IEC 7498-1) und der OSI-Application Layer-Struktur (ISO/IEC 9545).

1.2 Spezifikationen

Das hauptsächliche Ziel dieser Norm ist das Festlegen der Syntax und des Verhaltens des Application Layer-Protokolls, welches die in IEC 61158-5-2 beschriebenen Application-Layer-Dienste vermittelt.

Ein zweites Ziel ist das Bereitstellen von Migrationspfaden von bisherigen bestehenden industriellen Kommunikationsprotokollen.

1.3 Konformität

Diese Norm legt keine individuellen Implementierungen oder Produkte fest, auch beschränkt sie die Implementierungen von Application Layer-Einheiten innerhalb eines industriellen Automatisierungssystems nicht. Konformität wird erreicht durch das Implementieren dieser Application Layer-Protokoll-Spezifikation.

^{N1)} Die Nummerierung aus IEC 61158-6-2 wurde beibehalten.

3.3 Zusätzliche Begriffe

3.3.1

zuweisen

(en: allocate)

eine Ressource aus einem allgemeinen Bereich für die ausschließliche Nutzung durch eine spezifische Entität zuordnen

3.3.2

Anwendung

(en: application)

Funktion oder Datenstruktur, für die Daten verarbeitet oder erzeugt werden

3.3.3

Anwendungsobjekte

(en: application objects)

mehrere Objektklassen, die einen Nachrichtenaustausch über das Netzwerk und innerhalb des Netzwerkgerätes zur Laufzeit verwalten und bereitstellen

3.3.4

Attribut

(en: attribute)

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

Anmerkung 1 zum Begriff: 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.

3.3.5

Verhalten

(en: behavior)

Angabe, wie ein Objekt auf bestimmte Ereignisse reagiert

3.3.6

Best-Master-Clock-Algorithmus

BMC

(en: Best Master Clock Algorithm; BMCA)

Algorithmus, der von jedem Knoten durchgeführt wird, um die Uhr zu bestimmen, die die Master-Uhr in einem Subnetz und die Grandmaster-Uhr für die Domäne wird

Anmerkung 1 zum Begriff: Der Algorithmus vergleicht vorrangig „Priority1“, die Uhrenqualität, „Priority2“ und die Quellidentität, um den besten Master unter den verfügbaren Kandidaten zu ermitteln.

3.3.7

Grenzuhr

(en: boundary clock)

Uhr, die über mehrere PTP-Ports in einer Domäne verfügt und die in der Domäne verwendete Zeitskala aufrechterhält

Anmerkung 1 zum Begriff: Sie kann als Zeitquelle dienen, d. h. eine Master-Uhr sein, und mit einer anderen Uhr synchronisiert werden, d. h. eine Slave-Uhr sein.

[QUELLE: IEC 61588:2009, 3.1.3, modifiziert – zweiter Satz in eine Anmerkung geändert]

3.3.8

aufgerufen

(en: called)

Dienstbenutzer oder Dienstanbieter, der eine Meldeprimitive oder eine Anforderungs-APDU empfängt

3.3.9

aufrufend

(en: calling)

Dienstbenutzer oder Dienstanbieter, der eine Anforderungsprimitive oder eine Anforderungs-APDU initiiert

3.3.10

Klasse

(en: class)

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

Anmerkung 1 zum Begriff: Eine Klasse ist eine Verallgemeinerung eines Objekts; 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.

3.3.11

Klassenattribut

(en: class attribute)

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

3.3.12

Klassencode

(en: class code)

eindeutige Kennung, die jeder Objektklasse zugewiesen ist

3.3.13

klassenspezifischer Dienst

(en: class specific service)

durch eine bestimmte Objektklasse definierter Dienst, um eine erforderliche Funktion auszuführen, welche nicht als allgemeiner Dienst ausgeführt wird

Anmerkung 1 zum Begriff: Ein klassenspezifisches Objekt ist für die Objektklasse, die es definiert, eindeutig.

3.3.14

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

3.3.15

Uhr

(en: clock)

am PTP-Protokoll teilnehmender Knoten, der eine Messung der seit einem festgelegten Bezugszeitpunkt abgelaufenen Zeit bereitstellen kann

Anmerkung 1 zum Begriff: In IEC 61588:2009 sind drei Typen von Uhren festgelegt: Grenzuhren, transparente Uhren und gewöhnliche Uhren.

[QUELLE: IEC 61588:2009, 3.1.4, modifiziert – andere Anmerkung]

3.3.16

Kommunikationsobjekte

(en: communication objects)

Komponenten, die einen Austausch von Nachrichten über das Netzwerk zur Laufzeit bereitstellen und verwalten

BEISPIELE Verbindungsmanager-Objekt, UCMM-Objekt und Nachrichtenrouter-Objekt.

3.3.17

Verbindung

(en: connection)

logische Verknüpfung zwischen Anwendungsobjekten, die sich innerhalb eines Gerätes oder in unterschiedlichen Geräten befinden können

Anmerkung 1 zum Begriff: Verbindungen können entweder Punkt-zu-Punkt-Verbindungen oder Mehrpunktverbindungen sein.

3.3.18

Verbindungs-ID

CID

(en: connection ID; CID)

Kennung, die einer Übertragung im Zusammenhang mit einer bestimmten Verbindung zwischen Produzenten und Konsumenten zugewiesen ist und einen Namen für einen bestimmten Teil von Anwendungsinformationen bereitstellt

3.3.19

Verbindungspfad

(en: connection path)

Oktettenstrom, der das Anwendungsobjekt definiert, auf das sich eine Verbindungsinstanz bezieht

3.3.20

Verbindungspunkt

(en: connection point)

Puffer, der als Unterinstanz eines Assembly-Objekts dargestellt ist

3.3.21

konsumieren

(en: consume)

Vorgang des Empfangs von Daten von einem Produzenten

3.3.22

Konsument

(en: consumer)

Knoten oder Senke, der/die Daten von einem Produzenten empfängt

3.3.23

konsumierende Anwendung

(en: consuming application)

Anwendung, die Daten konsumiert

3.3.24

zyklisch

(en: cyclic)

sich regelmäßig wiederholend

3.3.25

Gerät

(en: device)

physische Hardware, die mit dem Verbindungszweig verbunden ist

Anmerkung 1 zum Begriff: Ein Gerät kann mehr als einen Knoten umfassen.

3.3.26

Geräteprofil

(en: device profile)

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

3.3.27

Domäne

(en: domain)

logische Gruppierung von Uhren, die sich mithilfe des Protokolls miteinander synchronisieren, die aber nicht zwangsläufig mit Uhren in einer anderen Domäne synchronisiert werden

[QUELLE: IEC 61588:2009, 3.1.7]

3.3.28

Endknoten

(en: end node)

produzierender oder konsumierender Knoten

3.3.29

Endpunkt

(en: end point)

eine der Kommunikationsentitäten, die zu einer Verbindung gehören

3.3.30

Bezugszeitpunkt

(en: epoch)

Anfang einer Zeitskala

[QUELLE: IEC 61588:2009, 3.1.9]

3.3.31

Fehler

(en: error)

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

3.3.32

Rahmen

(en: frame)

umgangssprachliches Synonym für DLPDU

3.3.33

Grandmaster-Uhr

(en: grandmaster clock)

innerhalb einer Domäne: Uhr, die als endgültige Zeitquelle für die Synchronisation der Uhren unter Verwendung des PTP-Protokolls dient

[QUELLE: IEC 61588:2009, 3.1.13]

3.3.34

Instanz

(en: instance)

tatsächliches physisches Vorkommen eines Objekts innerhalb einer Klasse, das eines von mehreren Objekten innerhalb derselben Objektklasse identifiziert

BEISPIEL „Kalifornien“ ist eine Instanz der Objektklasse „Staat“.

Anmerkung 1 zum Begriff: Die Benennungen „Objekt“, „Instanz“ und „Objektinstanz“ beziehen sich auf eine spezifische Instanz.

3.3.35

Instanzattribut

(en: instance attribute)

Attribut, das für eine Objektinstanz eindeutig ist und von der Objektklasse nicht gemeinsam genutzt wird

3.3.36**instanziiert**

(en: instantiated)

Objekt, das in einem Gerät erzeugt wurde

3.3.37**Interoperabilität**

(en: interoperability)

Fähigkeit von Entitäten der Benutzerschicht, mithilfe der Dienste des FAL koordinierte und kooperative Operationen auszuführen

3.3.38**Keeper**

(en: Keeper)

Objekt, das für die Verteilung von Konfigurationsdaten des Verbindungszweigs an alle Knoten auf dem Verbindungszweig verantwortlich ist

3.3.39**Little-Endian**

(en: little endian)

Modell der Speicherorganisation, bei dem das am wenigsten signifikante Oktett an der kleinsten Adresse gespeichert wird, oder Modell für die Übertragung, bei dem das niedrigstwertige Oktett zuerst übertragen wird

Anmerkung 1 zum Begriff: Native Datentypen vom Typ 2 werden in Little-Endian-Reihenfolge gesendet.

3.3.40**L-Paket**

Link-Paket

(en: Lpacket; Link packet)

Teil von Anwendungsinformationen, der eine Größe, ein Steuerungsoktett, ein tag und Daten des Data Link Layer enthält

Anmerkung 1 zum Begriff: Die Data Link Layer von Peers verwenden L-Pakete, um Dienstdateneinheiten an höhere Schichten im OSI-Stapel zu senden und um Dienstdateneinheiten von diesen zu empfangen.

3.3.41**Verwaltungsinformationen**

(en: management information)

über das Netzwerk zugängliche Informationen, die die Verwaltung des Betriebs des Feldbussystems, einschließlich des Application Layer, unterstützen

Anmerkung 1 zum Begriff: Zur Verwaltung gehören Funktionen, wie z. B. Steuerung, Überwachung und Diagnose.

3.3.42**Master-Uhr**

(en: master clock)

im Zusammenhang mit einem einzigen PTP-Kommunikationspfad: Uhr, die als Zeitquelle dient, mit der alle anderen Uhren auf diesem Pfad synchronisiert werden

[QUELLE: IEC 61588:2009, 3.1.17]

3.3.43**Member**

(en: member)

Teil eines Attributs, das als Element eines Feldes strukturiert ist

3.3.44**Nachrichtenrouter**

(en: Message Router)

Objekt innerhalb eines Knotens, das Nachrichten anforderungen an entsprechende Anwendungsobjekte verteilt

3.3.45

Mehrpunktverbindung

(en: multipoint connection)

Verbindung von einem Knoten zu vielen Knoten

Anmerkung 1 zum Begriff: Durch Mehrpunktverbindungen können Nachrichten von einem einzelnen Produzenten von vielen Konsumentenknoten empfangen werden.

3.3.46

Netzwerk

(en: network)

Menge von Knoten, die über ein bestimmtes Kommunikationsmedium miteinander verbunden sind, einschließlich zwischengeschalteter Repeater, Bridges, Router und Gateways unterer Schichten

3.3.47

Objekt

(en: object)

abstrakte Darstellung einer bestimmten Komponente innerhalb eines Gerätes, normalerweise eine Zusammenstellung zugehöriger Daten (in Form von Variablen) und Methoden (Prozeduren) für den Betrieb auf Basis dieser Daten, die eine klar definierte Schnittstelle und ein klar definiertes Verhalten aufweisen

3.3.48

objektspezifischer Dienst

(en: object specific service)

Dienst, der für die Objektklasse, die ihn definiert, eindeutig ist

3.3.49

gewöhnliche Uhr

(en: ordinary clock)

Uhr, die über einen einzigen PTP-Port in einer Domäne verfügt und die die in der Domäne verwendete Zeitskala aufrechterhält

Anmerkung 1 zum Begriff: Sie kann als Zeitquelle dienen, d. h. eine Master-Uhr sein, oder mit einer anderen Uhr synchronisiert werden, d. h. eine Slave-Uhr sein.

[QUELLE: IEC 61588:2009, 3.1.22, modifiziert – zweiter Satz in eine Anmerkung geändert]

3.3.50

Absender

(en: originator)

Client, der für den Aufbau eines Verbindungspfads zum Ziel verantwortlich ist

3.3.51

übergeordnete Uhr

(en: parent clock)

Master-Uhr, mit der eine Uhr synchronisiert wird

[QUELLE: IEC 61588:2009, 3.1.23]

3.3.52

Punkt-zu-Punkt-Verbindung

(en: point-to-point connection)

Verbindung, die zwischen genau zwei Anwendungsobjekten besteht

3.3.53

Precision Time Protocol

PTP

(en: Precision Time Protocol; PTP)

in IEC 61588:2009 festgelegtes Protokoll

Anmerkung 1 zum Begriff: Wird „PTP“ adjektivisch gebraucht, weist dies darauf hin, dass das modifizierte Nomen in IEC 61588:2009 festgelegt oder im Zusammenhang mit dieser Norm gedeutet wird.

[QUELLE: IEC 61588:2009, 3.1.28, modifiziert – zweiter Satz in eine Anmerkung geändert]

3.3.54

produzieren

(en: produce)

Vorgang des Sendens von Daten zum Empfang durch einen Konsumenten

3.3.55

Produzent

(en: producer)

Knoten, der für das Senden von Daten verantwortlich ist

3.3.56

PTP-Nachricht

(en: PTP message)

einer der in IEC 61588:2009 festgelegten Nachrichtentypen

[QUELLE: IEC 61588:2009, 3.1.33]

3.3.57

PTP-Port

(en: PTP port)

logischer Zugangspunkt einer Uhr für die PTP-Kommunikation mit dem Kommunikationsnetzwerk

[QUELLE: IEC 61588:2009, 3.1.35]

3.3.58

empfangend

(en: receiving)

Dienstbenutzer, der eine bestätigte oder eine unbestätigte Dienstprimitive empfängt, oder Dienstanbieter, der eine bestätigte APDU oder eine unbestätigte APDU empfängt

3.3.59

Ressource

(en: resource)

Verarbeitungs- oder Informationsfähigkeit eines Subsystems

3.3.60

sendend

(en: sending)

Dienstbenutzer, der eine bestätigte oder eine unbestätigte Dienstprimitive sendet, oder Dienstanbieter, der eine bestätigte APDU oder eine unbestätigte APDU sendet

3.3.61

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

3.3.62

Dienst

(en: service)

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

3.3.63

synchronisierte Uhren

(en: synchronized clocks)

(mit einer gewissen Messunsicherheit) zwei Uhren, die denselben Bezugszeitpunkt haben und bei denen Messungen des Zeitpunkts eines einzelnen Ereignisses zu einem beliebigen Zeitpunkt nicht mehr als die angegebene Messunsicherheit voneinander abweichen

[QUELLE: IEC 61588:2009, 3.1.40, modifiziert – umformuliert]

3.3.64

Systemzeit

(en: System Time)

absoluter Zeitwert nach der Definition der Zeitsynchronisation für die Kommunikationsprofilfamilie 2 im Kontext eines verteilten Systems, in dem alle Geräte über lokale Uhren verfügen, die mit einer gemeinsamen Master-Uhr synchronisiert werden

Anmerkung 1 zum Begriff: Im Rahmen der Kommunikationsprofilfamilie 2 entspricht die Systemzeit einem ganzzahligen 64-Bit-Wert in Nanosekunden, wobei sich der Wert „0“ auf das Datum „1970-01-01“ bezieht.

3.3.65

Ziel

(en: target)

Endknoten, mit dem eine Verbindung aufgebaut wird

3.3.66

temporärer Knoten

(en: temporary node)

transienter Knoten

3.3.67

Transaktions-ID

(en: transaction id)

Feld innerhalb eines UCMM-Headers, das eine Antwort der zugehörigen Anforderung zuordnet

3.3.68

transparente Uhr

(en: transparent clock)

Gerät, das die Zeit misst, die eine PTP-Ereignisnachricht durch das Gerät benötigt, und das diese Information den Uhren bereitstellt, die diese PTP-Ereignisnachricht empfangen

[QUELLE: IEC 61588:2009, 3.1.46, modifiziert – gekürzt]

3.3.69

Unconnected Message Manager

UCMM

(en: Unconnected Message Manager; UCMM)

Komponente innerhalb eines Knotens, die verbindungslose, explizite Nachrichten überträgt und empfängt und direkt an das Nachrichtenrouter-Objekt sendet

3.3.70

verbindungsloser Dienst

(en: unconnected service)

Nachrichtendienst, der nicht auf den Aufbau einer Verbindung zwischen Geräten angewiesen ist, bevor Informationen ausgetauscht werden können

3.3.71

Anbieter-ID

(en: vendor ID)

Kennzeichnung jedes Produktherstellers/Anbieters durch eine eindeutige Nummer

Anmerkung 1 zum Begriff: Anbieter-IDs werden von der ODVA, Inc. vergeben.

English Version

**Industrial communication networks - Fieldbus specifications -
Part 6-2: Application layer protocol specification - Type 2
elements
(IEC 61158-6-2:2014)**

Réseaux de communication industriels - Spécifications des
bus de terrain - Partie 6-2: Spécification du protocole de la
couche application - Éléments de type 2
(CEI 61158-6-2:2014)

Industrielle Kommunikationsnetze - Feldbusse - Teil 6-2:
Protokollspezifikation des Application Layer
(Anwendungsschicht) - Typ 2-Elemente
(IEC 61158-6-2:2014)

This European Standard was approved by CENELEC on 2014-09-23. 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, Former Yugoslav Republic of Macedonia, 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.



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

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Foreword

The text of document 65C/764/FDIS, future edition 3 of IEC 61158-6-2, 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 61158-6-2:2014.

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) 2015-06-23
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2017-09-23

This document supersedes EN 61158-6-2:2012.

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.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

Endorsement notice

The text of the International Standard IEC 61158-6-2:2014 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61131-3	NOTE	Harmonized as EN 61131-3.
IEC 61784-1:2014	NOTE	Harmonized as EN 61784-1:2014 (not modified).
IEC 61784-2:2014	NOTE	Harmonized as EN 61784-2:2014 (not modified).

CONTENTS

INTRODUCTION.....	13
1 Scope	14
1.1 General.....	14
1.2 Specifications	14
1.3 Conformance.....	14
2 Normative references	15
3 Terms, definitions, symbols, abbreviations and conventions	17
3.1 Terms and definitions from other ISO/IEC standards	17
3.2 Terms and definitions from IEC 61158-5-2	18
3.3 Additional terms and definitions	18
3.4 Abbreviations and symbols	25
3.5 Conventions	25
4 Abstract syntax.....	31
4.1 FAL PDU abstract syntax.....	31
4.2 Data abstract syntax specification	145
4.3 Encapsulation abstract syntax.....	150
5 Transfer syntax	168
5.1 Compact encoding.....	168
5.2 Data type reporting	176
6 Structure of FAL protocol state machines	183
7 AP-Context state machine	183
7.1 Overview	183
7.2 Connection object state machine.....	183
8 FAL service protocol machine (FSPM)	191
8.1 General.....	191
8.2 Primitive definitions	191
8.3 Parameters of primitives	196
8.4 FSPM state machines	196
9 Application relationship protocol machines (ARPMs)	196
9.1 General.....	196
9.2 Connection-less ARPM (UCMM)	197
9.3 Connection-oriented ARPMs (transports)	206
10 DLL mapping protocol machine 1 (DMPM 1).....	234
10.1 General.....	234
10.2 Link producer	234
10.3 Link consumer	235
10.4 Primitive definitions	235
10.5 DMPM state machine.....	237
10.6 Data-link Layer service selection.....	239
11 DLL mapping protocol machine 2 (DMPM 2).....	239
11.1 General.....	239
11.2 Mapping of UCMM PDUs	239
11.3 Mapping of transport class 0 and class 1 PDUs	247
11.4 Mapping of transport class 2 and class 3 PDU's	249

11.5 Mapping of transport classes 4 to 6.....	250
11.6 IGMP Usage.....	250
11.7 Quality of Service (QoS) for CP 2/2 messages	251
11.8 Management of an encapsulation session	254
12 DLL mapping protocol machine 3 (DMPM 3).....	255
Bibliography	256
Annex ZA (normative) Normative references to international publications with their corresponding European publications	257
Figure 1 – Attribute table format and terms.....	26
Figure 2 – Service request/response parameter	26
Figure 3 – Example of an STD	30
Figure 4 – Network connection parameters.....	53
Figure 5 – Time tick	56
Figure 6 – Connection establishment time-out	58
Figure 7 – Member ID/EX description (WORD)	70
Figure 8 – Transport Class Trigger attribute	102
Figure 9 – CP2/3_initial_comm_characteristics attribute format	106
Figure 10 – Segment type.....	114
Figure 11 – Port segment.....	115
Figure 12 – Logical segment encoding	117
Figure 13 – Extended network segment.....	122
Figure 14 – Symbolic segment encoding	123
Figure 15 – Encapsulation message.....	151
Figure 16 – FixedLengthBitString compact encoding bit placement rules.....	172
Figure 17 – Example compact encoding of a SWORD FixedLengthBitString	173
Figure 18 – Example compact encoding of a WORD FixedLengthBitString	173
Figure 19 – Example compact encoding of a DWORD FixedLengthBitString.....	173
Figure 20 – Example compact encoding of a LWORD FixedLengthBitString	173
Figure 21 – Example 1 of formal encoding of a structure type specification	178
Figure 22 – Example 2 of formal encoding of a structure type specification	178
Figure 23 – Example 3 of formal encoding of a handle structure type specification	179
Figure 24 – Example 4 of formal encoding of a handle structure type specification	179
Figure 25 – Example 5 of abbreviated encoding of a structure type specification.....	180
Figure 26 – Example 1 of formal encoding of an array type specification.....	181
Figure 27 – Example 2 of formal encoding of an array type specification.....	181
Figure 28 – Example 1 of abbreviated encoding of an array type specification	182
Figure 29 – Example 2 of abbreviated encoding of an array type specification	182
Figure 30 – I/O Connection object state transition diagram	183
Figure 31 – Bridged Connection object state transition diagram	187
Figure 32 – Explicit Messaging Connection object state transition diagram	189
Figure 33 – State transition diagram of UCMM client9	200
Figure 34 – State transition diagram of high–end UCMM server.....	201

Figure 35 – State transition diagram of low-end UCMM server	203
Figure 36 – Sequence diagram for a UCMM with one outstanding message	204
Figure 37 – Sequence diagram for a UCMM with multiple outstanding messages	205
Figure 38 – TPDU buffer	206
Figure 39 – Data flow diagram using a client transport class 0 and server transport class 0209	
Figure 40 – Sequence diagram of data transfer using transport class 0	209
Figure 41 – Class 0 client STD	210
Figure 42 – Class 0 server STD	211
Figure 43 – Data flow diagram using client transport class 1 and server transport class 1...212	
Figure 44 – Sequence diagram of data transfer using client transport class 1 and server transport class 1	213
Figure 45 – Class 1 client STD	214
Figure 46 – Class 1 server STD	216
Figure 47 – Data flow diagram using client transport class 2 and server transport class 2..218	
Figure 48 – Diagram of data transfer using client transport class 2 and server transport class 2 without returned data	219
Figure 49 – Sequence diagram of data transfer using client transport class 2 and server transport class 2 with returned data.....	220
Figure 50 – Class 2 client STD	221
Figure 51 – Class 2 server STD	223
Figure 52 – Data flow diagram using client transport class 3 and server transport class 3...226	
Figure 53 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 without returned data	227
Figure 54 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 with returned data.....	228
Figure 55 – Class 3 client STD	230
Figure 56 – Class 3 server STD	232
Figure 57 – Data flow diagram for a link producer and consumer	234
Figure 58 – State transition diagram for a link producer	238
Figure 59 – State transition diagram for a link consumer.....	239
Figure 60 – DS field in the IP header.....	252
Figure 61 – IEEE 802.1Q tagged frame	253
Table 1 – Get_Attribute_All response service rules	27
Table 2 – Example class level object/service specific response data of Get_Attribute_All	27
Table 3 – Example Get_Attribute_All data array method	28
Table 4 – Set_Attribute_All request service rules	29
Table 5 – Example Set_Attribute_All attribute ordering method	29
Table 6 – Example Set_Attribute_All data array method.....	29
Table 7 – State event matrix format.....	31
Table 8 – Example state event matrix.....	31
Table 9 – UCMM_PDU header format.....	35
Table 10 – UCMM command codes	35
Table 11 – Transport class 0 header	36

Table 12 – Transport class 1 header	36
Table 13 – Transport class 2 header	36
Table 14 – Transport class 3 header	36
Table 15 – Real-time data header – exclusive owner	37
Table 16 – Real-time data header– redundant owner	37
Table 17 – Forward_Open request format	41
Table 18 – Forward_Open_Good response format	42
Table 19 – Forward_Open_Bad response format	43
Table 20 – Large_Forward_Open request format	43
Table 21 – Large_Forward_Open_Good response format	44
Table 22 – Large_Forward_Open_Bad response format	45
Table 23 – Forward_Close request format	45
Table 24 – Forward_Close_Good response format	46
Table 25 – Forward_Close_Bad response format	46
Table 26 – Unconnected_Send request format	47
Table 27 – Unconnected_Send_Good response format	48
Table 28 – Unconnected_Send_Bad response format	48
Table 29 – Unconnected_Send request format (modified)	49
Table 30 – Unconnected_Send_Good response format (modified)	50
Table 31 – Unconnected_Send_Bad response format (modified)	50
Table 32 – Get_Connection_Data request format	50
Table 33 – Get_Connection_Data response format	51
Table 34 – Search_Connection_Data request format	51
Table 35 – Get_Connection_Owner request format	52
Table 36 – Get_Connection_Owner response format	52
Table 37 – Time-out multiplier	55
Table 38 – Time tick units	56
Table 39 – Encoded application path ordering	60
Table 40 – Transport class, trigger and Is_Server format	61
Table 41 – MR_Request_Header format	61
Table 42 – MR_Response_Header format	62
Table 43 – Structure of Get_Attribute_All_ResponsePDU body	63
Table 44 – Structure of Set_Attribute_All_RequestPDU body	63
Table 45 – Structure of Get_Attribute_List_RequestPDU body	63
Table 46 – Structure of Get_Attribute_List_ResponsePDU body	63
Table 47 – Structure of Set_Attribute_List_RequestPDU body	63
Table 48 – Structure of Set_Attribute_List_ResponsePDU body	64
Table 49 – Structure of Reset_RequestPDU body	64
Table 50 – Structure of Reset_ResponsePDU body	64
Table 51 – Structure of Start_RequestPDU body	64
Table 52 – Structure of Start_ResponsePDU body	64
Table 53 – Structure of Stop_RequestPDU body	65

Table 54 – Structure of Stop_ResponsePDU body	65
Table 55 – Structure of Create_RequestPDU body	65
Table 56 – Structure of Create_ResponsePDU body	65
Table 57 – Structure of Delete_RequestPDU body	65
Table 58 – Structure of Delete_ResponsePDU body	66
Table 59 – Structure of Get_Attribute_Single_ResponsePDU body	66
Table 60 – Structure of Set_Attribute_Single_RequestPDU body	66
Table 61 – Structure of Set_Attribute_Single_ResponsePDU body	66
Table 62 – Structure of Find_Next_Object_Instance_RequestPDU body	66
Table 63 – Structure of Find_Next_Object_Instance_ResponsePDU body	67
Table 64 – Structure of Apply_Attributes_RequestPDU body	67
Table 65 – Structure of Apply_Attributes_ResponsePDU body	67
Table 66 – Structure of Save_RequestPDU body	67
Table 67 – Structure of Save_ResponsePDU body	67
Table 68 – Structure of Restore_RequestPDU body	68
Table 69 – Structure of Restore_ResponsePDU body	68
Table 70 – Structure of Get_Member_ResponsePDU body	68
Table 71 – Structure of Set_Member_RequestPDU body	68
Table 72 – Structure of Set_Member_ResponsePDU body	68
Table 73 – Structure of Insert_Member_RequestPDU body	69
Table 74 – Structure of Insert_Member_ResponsePDU body	69
Table 75 – Structure of Remove_Member_ResponsePDU body	69
Table 76 – Common structure of _Member_RequestPDU body (basic format)	70
Table 77 – Common structure of _Member_ResponsePDU body (basic format)	70
Table 78 – Common structure of _Member_RequestPDU body (extended format)	71
Table 79 – Common structure of _Member_ResponsePDU body (extended format)	71
Table 80 – Extended Protocol ID	71
Table 81 – Structure of _Member_RequestPDU body (Multiple Sequential Members)	72
Table 82 – Structure of _Member_ResponsePDU body (Multiple Sequential Members)	72
Table 83 – Structure of _Member_RequestPDU body (International String Selection)	72
Table 84 – Structure of _Member_ResponsePDU body (International String Selection)	73
Table 85 – Structure of Group_Sync_RequestPDU body	73
Table 86 – Structure of Group_Sync_ResponsePDU body	73
Table 87 – Identity object class attributes	73
Table 88 – Identity object instance attributes	74
Table 89 – Identity object bit definitions for status instance attribute	75
Table 90 – Default values for extended device status field (bits 4 to 7) of status instance attribute	75
Table 91 – Class level object/service specific response data of Get_Attribute_All	76
Table 92 – Instance level object/service specific response data of Get_Attribute_All	76
Table 93 – Object-specific parameter for Reset	76
Table 94 – Reset service parameter values	77

Table 95 – Message Router object class attributes	77
Table 96 – Message Router object instance attributes	77
Table 97 – Class level object/service specific response data of Get_Attribute_All	78
Table 98 – Instance level object/service specific response data of Get_Attribute_All	78
Table 99 – Structure of Symbolic_Translation_RequestPDU body	78
Table 100 – Structure of Symbolic_Translation_ResponsePDU body	78
Table 101 – Object specific status for Symbolic_Translation service	78
Table 102 – Assembly object class attributes	79
Table 103 – Assembly object instance attributes	79
Table 104 – Assembly Instance ID ranges	79
Table 105 – Acknowledge Handler object class attributes	80
Table 106 – Acknowledge Handler object instance attributes	80
Table 107 – Structure of Add_AckData_Path_RequestPDU body	81
Table 108 – Structure of Remove_AckData_Path_RequestPDU body	81
Table 109 – Time Sync object class attributes	81
Table 110 – Time Sync object instance attributes	82
Table 111 – ClockIdentity encoding for different network implementations	86
Table 112 – ClockClass values	86
Table 113 – TimeAccuracy values	86
Table 114 – TimePropertyFlags bit values	87
Table 115 – TimeSource values	87
Table 116 – Types of Clock	87
Table 117 – Network protocol to PortPhysicalAddressInfo mapping	88
Table 118 – Parameter object class attributes	88
Table 119 – Parameter Class Descriptor bit values	89
Table 120 – Parameter object instance attributes	89
Table 121 – Semantics of Descriptor Instance attribute	90
Table 122 – Minimum and Maximum Value semantics	91
Table 123 – Scaling Formula attributes	92
Table 124 – Scaling links	92
Table 125 – Class level object/service specific response data of Get_Attribute_All	93
Table 126 – Instance level object/service specific response data of Get_Attribute_All (Parameter object stub)	94
Table 127 – Instance level object/service specific response data of Get_Attribute_All (full Parameter object)	94
Table 128 – Structure of Get_Enum_String_RequestPDU body	95
Table 129 – Structure of Get_Enum_String_ResponsePDU body	96
Table 130 – Enumerated strings Type versus Parameter data type	96
Table 131 – Connection Manager object class attributes	96
Table 132 – Connection Manager object instance attributes	97
Table 133 – Class level object/service specific response data of Get_Attribute_All	97
Table 134 – Instance level object/service specific response data of Get_Attribute_All	98
Table 135 – Instance level object/service specific request data of Set_Attribute_All	99

Table 136 – Connection object class attributes	99
Table 137 – Connection object instance attributes	100
Table 138 – Values assigned to the state attribute	101
Table 139 – Values assigned to the instance_type attribute	101
Table 140 – Possible values within Direction Bit	102
Table 141 – Possible values within Production Trigger Bits	103
Table 142 – Possible values within Transport Class Bits	103
Table 143 – TransportClass_Trigger attribute values summary	104
Table 144 – Transport Class 0 client behavior summary	104
Table 145 – Transport Class 1, 2 and 3 client behavior summary	105
Table 146 – Values defined for the CP2/3_produced_connection_id attribute	105
Table 147 – Values defined for the CP2/3_consumed_connection_id attribute	105
Table 148 – Values for the Initial Production Characteristics nibble	106
Table 149 – Values for the Initial Consumption Characteristics nibble	107
Table 150 – Values for the watchdog_timeout_action	110
Table 151 – Structure of Connection_Bind_RequestPDU body	112
Table 152 – Object specific status for Connection_Bind service	112
Table 153 – Structure of Producing_Application_Lookup_RequestPDU body	112
Table 154 – Structure of Producing_Application_Lookup_ResponsePDU body	112
Table 155 – Producing_Application_Lookup Service status codes	113
Table 156 – Possible port segment examples	115
Table 157 – TCP/IP link address examples	116
Table 158 – Extended Logical Type	117
Table 159 – Electronic key segment format	119
Table 160 – Logical segments examples	120
Table 161 – Network segments	120
Table 162 – Extended subtype definitions	122
Table 163 – Symbolic segment examples	123
Table 164 – Data segment	124
Table 165 – ANSI_Extended_Symbol segment	124
Table 166 – Addressing categories	127
Table 167 – Class code ID ranges	127
Table 168 – Attribute ID ranges	128
Table 169 – Service code ranges	128
Table 170 – Class codes	128
Table 171 – Reserved class attributes for all object class definitions	129
Table 172 – Common services list	130
Table 173 – Message Router object specific services list	130
Table 174 – Acknowledge Handler object specific services list	131
Table 175 – Parameter object specific services list	131
Table 176 – Services specific to Connection Manager	131
Table 177 – Services specific to Connection object	131

Table 178 – Device type numbering	132
Table 179 – Connection Manager service request error codes	133
Table 180 – General status codes	141
Table 181 – Extended status code for a general status of "Key Failure in path	143
Table 182 – Identity object status codes.....	144
Table 183 – Encapsulation header	151
Table 184 – Encapsulation command codes	152
Table 185 – Encapsulation status codes	153
Table 186 – Nop request encapsulation header	154
Table 187 – RegisterSession request encapsulation header	155
Table 188 – RegisterSession request data portion	155
Table 189 – RegisterSession reply encapsulation header	155
Table 190 – RegisterSession reply data portion.....	156
Table 191 – UnRegisterSession request encapsulation header	157
Table 192 – ListServices request encapsulation header.....	157
Table 193 – ListServices reply encapsulation header.....	157
Table 194 – ListServices reply data portion	158
Table 195 – Communications capability flags	158
Table 196 – ListIdentity request encapsulation header.....	159
Table 197 – ListIdentity reply encapsulation header	160
Table 198 – ListIdentity reply data portion (successful).....	160
Table 199 – CPF 2 identity item	160
Table 200 – ListInterfaces request encapsulation header.....	161
Table 201 – ListInterfaces reply encapsulation header.....	161
Table 202 – SendRRData request encapsulation header	162
Table 203 – SendRRData request data portion.....	162
Table 204 – SendRRData reply encapsulation header	163
Table 205 – SendUnitData request encapsulation header	163
Table 206 – SendUnitData request data portion	163
Table 207 – Common packet format.....	164
Table 208 – CPF item format	164
Table 209 – Item Type ID numbers	164
Table 210 – Null address item.....	165
Table 211 – Connected address item	165
Table 212 – Sequenced address item	166
Table 213 – Unconnected data item	166
Table 214 – Connected data item.....	166
Table 215 – Sockaddr info items	167
Table 216 – Usage of CPF items.....	168
Table 217 – BOOLEAN encoding	169
Table 218 – Example compact encoding of a BOOL value	169
Table 219 – Encoding of SignedInteger values	169

Table 220 – Example compact encoding of a SignedInteger value	169
Table 221 – UnsignedInteger values	170
Table 222 – Example compact encoding of an UnsignedInteger	170
Table 223 – FixedLengthReal values.....	170
Table 224 – Example compact encoding of a REAL value	170
Table 225 – Example compact encoding of a LREAL value	170
Table 226 – FixedLengthReal values.....	171
Table 227 – STRING value	171
Table 228 – STRING2 value.....	171
Table 229 – STRINGN value	171
Table 230 – SHORT_STRING value	172
Table 231 – Example compact encoding of a STRING value	172
Table 232 – Example compact encoding of STRING2 value	172
Table 233 – SHORT_STRING type.....	172
Table 234 – Example compact encoding of a single dimensional ARRAY	174
Table 235 – Example compact encoding of a multi-dimensional ARRAY	174
Table 236 – Example compact encoding of a STRUCTURE	175
Table 237 – Identification codes and descriptions of elementary data types	176
Table 238 – Identification codes and descriptions of constructed data types.....	177
Table 239 – Formal structure encoding definition.....	177
Table 240 – Formal structure with handles encoding definition.....	178
Table 241 – Abbreviated structure encoding definition	179
Table 242 – Formal array encoding definition	180
Table 243 – Abbreviated array encoding definition.....	182
Table 244 – I/O Connection state event matrix	184
Table 245 – Bridged Connection state event matrix	188
Table 246 – Explicit Messaging Connection state event matrix.....	189
Table 247 – Primitives issued by FAL user to FSPM	192
Table 248 – Primitives issued by FAL user to FSPM	192
Table 249 – Primitives issued by FSPM to FAL user	195
Table 250 – Parameters used with primitives exchanged between FAL user and FSPM.....	196
Table 251 – Primitives issued by FSPM to ARPM	198
Table 252 – Primitives issued by ARPM to FSPM	198
Table 253 – Parameters used with primitives exchanged between FSPM and ARPM	199
Table 254 – UCMM client states.....	199
Table 255 – State event matrix of UCMM client	200
Table 256 – High-end UCMM server states.....	201
Table 257 – State event matrix of high-end UCMM server.....	202
Table 258 – Low-end UCMM server states	203
Table 259 – State event matrix of low-end UCMM server	204
Table 260 – Notification	206
Table 261 – Transport classes	207

Table 262 – Primitives issued by FSPM to ARPM	207
Table 263 – Primitives issued by ARPM to FSPM	208
Table 264 – Parameters used with primitives exchanged between FSPM and ARPM	208
Table 265 – Class 0 transport client states	210
Table 266 – Class 0 client SEM	210
Table 267 – Class 0 transport server states	211
Table 268 – Class 0 server SEM	212
Table 269 – Class 1 transport client states	214
Table 270 – Class 1 client SEM	215
Table 271 – Class 1 transport server states	216
Table 272 – Class 1 server SEM	217
Table 273 – Class 2 transport client states	221
Table 274 – Class 2 client SEM	222
Table 275 – Class 2 transport server states	223
Table 276 – Class 2 server SEM	224
Table 277 – Class 3 transport client states	229
Table 278 – Class 3 client SEM	230
Table 279 – Class 3 transport server states	231
Table 280 – Class 3 server SEM	233
Table 281 – Primitives issued by ARPM to DMPM	235
Table 282 – Primitives issued by DMPM to ARPM	235
Table 283 – Parameters used with primitives exchanged between ARPM and DMPM	235
Table 284 – Primitives exchanged between data-link layer and DMPM	236
Table 285 – Parameters used with primitives exchanged between DMPM and Data-link	236
Table 286 – Selection of connection ID	237
Table 287 – Link producer states	237
Table 288 – State event matrix of link producer	238
Table 289 – Link consumer states	238
Table 290 – State event matrix of link consumer	239
Table 291 – UCMM request	240
Table 292 – UCMM reply	241
Table 293 – Network Connection ID selection	242
Table 294 – Sockaddr Info usage	244
Table 295 – Example multicast assignments	247
Table 296 – UDP data format for class 0 and class 1	248
Table 297 – Transport class 2 and class 3 connected data	249
Table 298 – Default DSCP and IEEE 802.1D mapping	253

INTRODUCTION

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

The application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this 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 6-2: 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

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

This standard specifies the protocol of the Type 2 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this 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.

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 documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE All parts of the IEC 61158 series, as well as IEC 61784-1 and IEC 61784-2 are maintained simultaneously. Cross-references to these documents within the text therefore refer to the editions as dated in this list of normative references.

IEC 61158-1:2014, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-2:2014, *Industrial communication networks – Fieldbus specifications – Part 3-2: Data-link layer service definition – Type 2 elements*

IEC 61158-4-2:2014, *Industrial communication networks – Fieldbus specifications – Part 4-2: Data-link layer protocol specification – Type 2 elements*

IEC 61158-5-2:2014, *Industrial communication networks – Fieldbus specifications – Part 5-2: Application layer service definition – Type 2 elements*

IEC 61588:2009, *Precision clock synchronization protocol for networked measurement and control systems*

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

IEC 61800-7-202, *Adjustable speed electrical power drive systems – Part 7-202: Generic interface and use of profiles for power drive systems – Profile type 2 specification*

IEC 62026-3:2008, *Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet*

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

ISO/IEC 8802-3, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

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

ISO/IEC 8825-1, *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 639-2, *Codes for the representation of names of languages – Part 2: Alpha-3 code*

ISO 11898:1993¹, *Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication*

IEEE 802.1D-2004, *IEEE standard for local and metropolitan area networks – Media Access Control (MAC) bridges*, available at <http://www.ieee.org>

IEEE 802.1Q-2005¹, *IEEE standard for local and metropolitan area networks – Virtual bridged local area networks*, available at <http://www.ieee.org>

IEEE 802.3-2008: *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, available at <http://www.ieee.org>

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

IETF RFC 1035, *Domain Names – Implementation and Specification*, available at <http://www.ietf.org>

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

IETF RFC 1117, *Internet Numbers*, available at <http://www.ietf.org>

IETF RFC 1122, *Requirements for Internet Hosts – Communication Layers*, available at <http://www.ietf.org>

IETF RFC 1759, *Printer MIB*, available at <http://www.ietf.org>

IETF RFC 2236, *Internet Group Management Protocol, Version 2*, available at <http://www.ietf.org>

IETF RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, available at <http://www.ietf.org>

IETF RFC 2475, *An Architecture for Differentiated Services*, available at <http://www.ietf.org>

IETF RFC 2597, *Assured Forwarding PHB Group*, available at <http://www.ietf.org>

IETF RFC 2873, *TCP Processing of the IPv4 Precedence Field*, available at <http://www.ietf.org>

IETF RFC 3140, *Per Hop Behavior Identification Codes*, available at <http://www.ietf.org>

IETF RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, available at <http://www.ietf.org>

IETF RFC 3376, *Internet Group Management Protocol, Version 3*, available at <http://www.ietf.org>

IETF RFC 4594, *Configuration Guidelines for DiffServ Service Classes*, available at <http://www.ietf.org>

¹ A newer edition of this standard has been published, but only the cited edition applies.

3 Terms, definitions, symbols, abbreviations and conventions

For the purposes of this document, the following terms, definitions, symbols, abbreviations and conventions 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 8824-1

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

3.1.4 Terms and definitions from ISO/IEC 8825-1

- a) encoding (of a data value)
- b) data value
- c) identifier octets (the singular form is used in this 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
- d) server

3.3 Additional terms and definitions

3.3.1

allocate

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

3.3.2

application

function or data structure for which data is consumed or produced

3.3.3

application objects

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

3.3.4

attribute

description of an externally visible characteristic or feature of an object

Note 1 to entry: The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behavior of an object. Attributes are divided into class attributes and instance attributes.

3.3.5

behavior

indication of how an object responds to particular events

3.3.6

Best Master Clock Algorithm

BMCA

algorithm performed by each node to determine the clock that will become the master clock on a subnet and the grandmaster clock for the domain

Note 1 to entry: The algorithm primarily compares priority1, clock quality, priority2, and source identity to determine the best master among available candidates.

3.3.7

boundary clock

clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain

Note 1 to entry: It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock.

[SOURCE: IEC 61588:2009, 3.1.3, modified – second sentence changed to a Note]

3.3.8

called

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

3.3.9

calling

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

3.3.10

class

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

Note 1 to entry: A class is a generalization of an object; a template for defining variables and methods. All objects in a class are identical in form and behavior, but usually contain different data in their attributes.

3.3.11

class attribute

attribute that is shared by all objects within the same class

3.3.12

class code

unique identifier assigned to each object class

3.3.13

class specific service

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

Note 1 to entry: A class specific object is unique to the object class which defines it.

3.3.14

client

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

3.3.15

clock

node participating in the Precision Time Protocol (PTP) that is capable of providing a measurement of the passage of time since a defined epoch

Note 1 to entry: There are three types of clocks in IEC 61588:2009, boundary, transparent and ordinary clocks.

[SOURCE: IEC 61588:2009, 3.1.4, modified – different Note]

3.3.16

communication objects

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

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

3.3.17

connection

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

Note 1 to entry: Connections may be either point-to-point or multipoint.

3.3.18

connection ID

CID

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

3.3.19

connection path

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

3.3.20

connection point

buffer which is represented as a subinstance of an Assembly object

3.3.21

consume

act of receiving data from a producer

3.3.22

consumer

node or sink that is receiving data from a producer

3.3.23

consuming application

application that consumes data

3.3.24

cyclic

repetitive in a regular manner

3.3.25

device

physical hardware connected to the link

Note 1 to entry: A device may contain more than one node.

3.3.26

device profile

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

3.3.27

domain

logical grouping of clocks that synchronize to each other using the protocol, but that are not necessarily synchronized to clocks in another domain

[SOURCE: IEC 61588:2009, 3.1.7]

3.3.28

end node

producing or consuming node

3.3.29

end point

one of the communicating entities involved in a connection

3.3.30

epoch

origin of a time scale

[SOURCE: IEC 61588:2009, 3.1.9]

3.3.31

error

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

3.3.32

frame

denigrated synonym for DLPDU

3.3.33

grandmaster clock

within a domain, clock that is the ultimate source of time for clock synchronization using the PTP protocol

[SOURCE: IEC 61588:2009, 3.1.13]

3.3.34

instance

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

EXAMPLE California is an instance of the object class state.

Note 1 to entry: The terms object, instance, and object instance are used to refer to a specific instance.

3.3.35

instance attribute

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

3.3.36

instantiated

object that has been created in a device

3.3.37

interoperability

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

3.3.38

Keeper

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

3.3.39

little endian

model of memory organization which stores the least significant octet at the lowest address, or for transfer, which transfers the lowest order octet first

Note 1 to entry: Native Type 2 data types are sent in little endian order.

3.3.40

Lpacket

Link packet

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

Note 1 to entry: Peer data-link layers use Lpackets to send and receive service data units from higher layers in the OSI stack.

3.3.41

management information

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

Note 1 to entry: Managing includes functions such as controlling , monitoring, and diagnosing.

3.3.42

master clock

in the context of a single Precision Time Protocol (PTP) communication path, clock that is the source of time to which all other clocks on that path synchronize

[SOURCE: IEC 61588:2009, 3.1.17]

3.3.43

member

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

3.3.44

Message Router

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

3.3.45

multipoint connection

connection from one node to many

Note 1 to entry: Multipoint connections allow messages from a single producer to be received by many consumer nodes.

3.3.46

network

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

3.3.47

object

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

3.3.48

object specific service

service unique to the object class which defines it

3.3.49

ordinary clock

clock that has a single Precision Time Protocol (PTP) port in a domain and maintains the timescale used in the domain

Note 1 to entry: It may serve as a source of time, i.e., be a master clock, or may synchronize to another clock, i.e., be a slave clock.

[SOURCE: IEC 61588:2009, 3.1.22, modified – second sentence changed to a Note]

3.3.50

originator

client responsible for establishing a connection path to the target

3.3.51

parent clock

master clock to which a clock is synchronized

[SOURCE: IEC 61588:2009, 3.1.23]

3.3.52

point-to-point connection

connection that exists between exactly two application objects

3.3.53

Precision Time Protocol

PTP

protocol defined by IEC 61588:2009

Note 1 to entry: As an adjective, it indicates that the modified noun is specified in or interpreted in the context of IEC 61588:2009.

[SOURCE: IEC 61588:2009, 3.1.28, modified – second sentence changed to a Note]

3.3.54

produce

act of sending data to be received by a consumer

3.3.55

producer

node that is responsible for sending data

3.3.56

PTP message

one of the message types defined in IEC 61588:2009

[SOURCE: IEC 61588:2009, 3.1.33]

3.3.57

PTP port

logical access point of a clock for PTP communications to the communications network

[SOURCE: IEC 61588:2009, 3.1.35]

3.3.58

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

3.3.59

resource

processing or information capability of a subsystem

3.3.60

sending

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

3.3.61

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

3.3.62

service

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

3.3.63

synchronized clocks

(to a specified uncertainty) two clocks which have the same epoch and for which measurements of the time of a single event at an arbitrary time differ by no more than the specified uncertainty

[SOURCE: IEC 61588:2009, 3.1.40, modified – reworded]

3.3.64

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 1 to entry: 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.

3.3.65

target

end-node to which a connection is established

3.3.66

temporary node

transient node

3.3.67

transaction id

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

3.3.68

transparent clock

device that measures the time taken for a Precision Time Protocol (PTP) event message to transit the device and provides this information to clocks receiving this PTP event message

[SOURCE: IEC 61588:2009, 3.1.46, modified – truncated]

3.3.69

Unconnected Message Manager

UCMM

component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object

3.3.70

unconnected service

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

3.3.71

vendor ID

identification of each product manufacturer/vendor by a unique number

Note 1 to entry: Vendor IDs are assigned by ODVA, Inc.

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 IETF RFC 1112, IETF RFC 2236)
IP	Internet Protocol (see IETF RFC 791)
IPv4	Internet Protocol version 4 (see IETF RFC 791)
IPv6	Internet Protocol version 6 (see IETF RFC 791)
OSI	open systems interconnection (see ISO/IEC 7498-1)
PDU	protocol data unit
PHB	per-hop behavior
PTP	Precision Time Protocol (see IEC 61588:2009)
QoS	quality of service
Rcv	receive
Rx	receive
SDU	service data unit
SEM	state event matrix
STD	state transition diagram, used to describe object behavior
TCP	Terminal Control Protocol (see IETF RFC 793)
ToS	type of service
TPDU	transport protocol data unit
Tx	transmit
UDP	User Datagram Protocol (see IETF 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.

Bold font is used in this standard to highlight parameter names or important requirement elements from surrounding text. Italic font is also used in Tables and Notes for that purpose for better visibility.

3.5.2 Attribute specification

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

Attribute ID	Name	Data type	Semantics of values
--------------	------	-----------	---------------------

Figure 1 – Attribute table format and terms

The **Attribute ID** shall be a unique integer identification value assigned to an attribute. The valid ranges for Attribute IDs shall be as specified in 4.1.10.1.3. The Attribute ID shall identify the particular attribute being accessed.

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

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

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

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

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

3.5.3 Common services

3.5.3.1 Service_PDU definitions

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

Name	Type	Semantics of values
------	------	---------------------

Figure 2 – Service request/response parameter

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

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

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

3.5.3.2 Get_Attribute_All response

3.5.3.2.1 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;
- 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	<p>If the definition of the Get_Attribute_All response includes optional attributes, then default values shall be specified in the response description. Optional attributes at the end of the list that are not implemented may be omitted in the response data. Optional attributes in the middle of the list that are not implemented shall be included in the response data, and set to specified default values.</p> <p>If any of the following optional attributes are included in an object specification, but not supported in the implementation of the object, then the following shall be adhered to:</p> <ul style="list-style-type: none"> – 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 for objects specified in this standard 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.

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)							

3.5.3.2.2 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.

3.5.3.3 Set_Attribute_All request

3.5.3.3.1 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.

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	Default values shall be specified for all optional attributes that are not implemented. If an implementation does not support an optional attribute, it shall accept the specified default value for the unsupported attribute.
3	If new settable attributes are added to an existing object, those attributes shall be added to the end of the request attribute list or data array.
4	Whichever method is used to specify the response, it shall be done in such a way as to be unambiguous, including rules to deal with variable length fields and padding.
5	The Set_Attribute_All request response for objects specified in this standard 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)							

3.5.3.3.2 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 recognized. 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 0x15 (too much data).

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

3.5.4 State machine conventions

3.5.4.1 General

State changes 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.

3.5.4.2 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.

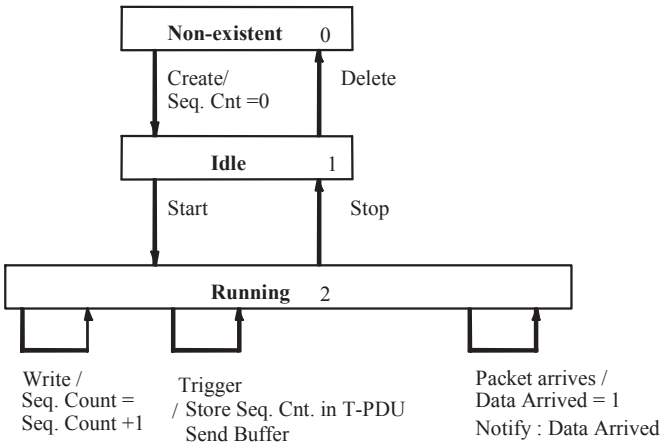


Figure 3 – Example of an STD

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

3.5.4.3 State Event Matrix

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

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

State event matrix format is as shown in Table 7.

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.			

3.5.4.4 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.

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 object, based on the corresponding connection ID. Connected messages sent to the Message Router will be forwarded to the appropriate internal object for execution.

4.1.2 PDU structure

4.1.2.1 UCMM_PDU structure

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

OM_Service shall be either OM_Request or OM_Response.

OM_Request shall consist of MR_Request_Header and a Service_RequestPDU.

Service_RequestPDU shall be one of the following:

- Get_Attribute_Single_RequestPDU
- Get_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
- Restore_RequestPDU
- Get_Member_RequestPDU
- Set_Member_RequestPDU
- Insert_Member_RequestPDU
- Remove_Member_RequestPDU
- Group_Sync_RequestPDU
- Add_AckData_Path_RequestPDU
- Remove_AckData_Path_RequestPDU
- Get_Enum_String_RequestPDU
- Symbolic_Translation_RequestPDU
- 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
- Get_Member_ResponsePDU
- Set_Member_ResponsePDU
- Insert_Member_ResponsePDU
- Remove_Member_ResponsePDU
- Group_Sync_ResponsePDU
- Add_AckData_Path_ResponsePDU
- Remove_AckData_Path_ResponsePDU
- Get_Enum_String_ResponsePDU
- Symbolic_Translation_ResponsePDU
- 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
- Search_Connection_Data_RequestPDU
- Get_Connection_Owner_RequestPDU

CM_Response consists of MR_Response_Header and a CM_ResponsePDU.

CM_ResponsePDU shall be one of the following:

- Forward_Open_ResponsePDU
- Forward_Close_ResponsePDU
- Large_Forward_Open_ResponsePDU
- Unconnected_Send_ResponsePDU
- Get_Connection_Data_ResponsePDU
- Search_Connection_Data_ResponsePDU
- Get_Connection_Owner_ResponsePDU

4.1.2.2 Transport_PDU structure

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

OM_Service is defined in 4.1.2.1 above.

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

4.1.3 UCMM_PDUs

Subclause 4.1.3 defines the requirements for UCMM PDU's when using a Type 2 data-link layer.

All UCMM_PDUs shall be sent and received using fixed tag 0x83 or Management tag 0x88. When a device becomes powered, the UCMM shall invoke the `DLL_enable_fixed_request` service of the data-link layer to request that fixed tag 0x83 or 0x88 Lpackets be routed to the UCMM. All sending and receiving of TPDUs shall use the `DLL_fixed_request.req` and `DLL_fixed_request.ind` service primitives of the data-link layer, respectively. The packet parameter of these services shall be prefixed with the following header to form the Transport PDU before sending to the data-link layer. The format of the UCMM_PDU Header is shown in Table 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
4	request with no acknowledge and no response
5	acknowledge a response
6	response which shall not retry (no acknowledge)
7	request with retry until response (no acknowledge)
8	Request which shall not retry and shall cause a code 6 response
9 to 255	Reserved

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

NOTE Using ANSI C precedence rules, the number of 0,125 ms ticks is $(8 \mid \text{timeout} \& 7) \ll (\text{timeout} \gg 3 \& 31)$.

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

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

Only one message shall be outstanding for a given RECORD.

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

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

4.1.4 Transport_Headers

4.1.4.1 General

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

4.1.4.2 Class 0 (null or base)

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

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)

NOTE Subclause contents from the previous edition have been obsoleted.

4.1.4.7 Listen only O⇒T data format

The O⇒T connection shall use a specific heartbeat format (no 32-bit header, 0 octets of application data).

4.1.4.8 Input only O⇒T data format

The O⇒T connection shall use a specific heartbeat format (no 32-bit header, 0 octets of application data).

4.1.4.9 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 15.

Table 15 – Real-time data header – exclusive owner

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

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

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

4.1.4.10 Redundant owner O⇒T data format

4.1.4.10.1 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 16.

Table 16 – Real-time data header– redundant owner

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

4.1.4.10.2 Run_idle flag

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

4.1.4.10.3 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.

4.1.4.10.4 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.

4.1.4.10.5 Determining the owning connection

The originator applications shall determine among themselves which originator application has the owning connection. The owning connection shall be determined by the originator application that sets its COO flag. In situations where multiple originator applications have their COO flag set or where no originator connections have their COO flag set, the following rules shall be applied by the target transport to determine the owning connection.

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

4.1.4.10.6 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