

DIN EN 61158-5-15

ICS 35.100.70; 35.160

Ersatz für
DIN EN 61158-5-15:2008-09
Siehe Anwendungsbeginn

**Industrielle Kommunikationsnetze –
Feldbusse –
Teil 5-15: Dienstfestlegungen des Application Layer
(Anwendungsschicht) –
Typ 15-Elemente
(IEC 61158-5-15:2010);
Englische Fassung EN 61158-5-15:2012, nur auf CD-ROM**

Industrial communication networks –
Fieldbus specifications –
Part 5-15: Application layer service definition –
Type 15 elements
(IEC 61158-5-15:2010);
English version EN 61158-5-15:2012, only on CD-ROM

Réseaux de communication industriels –
Spécifications des bus de terrain –
Partie 5-15: Définition des services des couches d'application –
Éléments de type 15
(CEI 61158-5-15:2010);
Version anglaise EN 61158-5-15:2012, seulement en CD-ROM

Gesamtumfang 147 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Anwendungsbeginn

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

Für DIN EN 61158-5-15:2008-09 gilt eine Übergangsfrist bis zum 2015-03-28.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN IEC 61158-500:2009-06.

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

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

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

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

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

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

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

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

Da sich die Benutzer der vorliegenden Norm der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 61158-5-15 veröffentlicht. Zu deren Abschnitt 3, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als informativer Nationaler Anhang NA der vorliegenden Norm hinzugefügt. Für viele der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Diese Norm steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Einführung

Ein Feldbus ist ein digitaler serieller Datenbus für die Kommunikation zwischen Geräten der industriellen Automatisierungstechnik wie beispielsweise Messeinrichtungen, Stellgliedern, Reglern und speicherprogrammierbaren Steuerungen. Die Normen der Reihe DIN EN 61158 legen eine Anzahl von Feldbus-Protokolltypen fest. Die vorliegende Norm legt die Dienste der Anwendungsschicht für Typ 15 fest.

Änderungen

Gegenüber DIN EN 61158-5-15:2008-09 wurden folgende Änderungen vorgenommen:

- a) redaktionelle Verbesserungen.

Frühere Ausgaben

DIN V 19245-1: 1988-01
 DIN 19245-1: 1991-04
 DIN 19245-2: 1991-04
 DIN EN 50170/1: 1997-07
 DIN EN 50170/2: 1997-07
 DIN EN 50170/3: 1997-07
 DIN EN 50254: 1999-07
 DIN EN 50170/A2: 2000-08
 DIN EN 50170/A1: 2002-11
 DIN EN 50170/A3: 2002-11
 DIN EN 61158-5: 2005-05
 DIN EN 61158-5-15: 2008-09

Nationaler Anhang NA (informativ)

3.1.5 Begriffe aus IEC/TR 61158-1

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

3.1.5.1

Anwendung

(en: application)

Funktion oder Datenstruktur, für die Daten verarbeitet oder erzeugt werden

3.1.5.2

Interoperabilität auf Anwendungsschicht

(en: application layer interoperability)

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

3.1.5.3

Anwendungsobjekt

(en: application object)

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

ANMERKUNG Es können mehrere Typen von Anwendungsobjektklassen definiert werden.

3.1.5.4

Anwendungsprozess

(en: application process)

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

3.1.5.5

Anwendungsprozesskennung

(en: application process identifier)

unterscheidet mehrere in einem Gerät verwendete Anwendungsprozesse

3.1.5.6

Anwendungsprozessobjekt

(en: application process object)

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

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

3.1.5.7

Klasse von Anwendungsprozessobjekten

(en: application process object class)

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

3.1.5.8

Anwendungsbeziehung

(en: application relationship)

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

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

3.1.5.9

Endpunkt der Anwendungsbeziehung

(en: application relationship endpoint)

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

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

3.1.5.10

Anwendungsdienstelement

(en: application service element)

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

3.1.5.11

Attribut

(en: attribute)

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

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

3.1.5.12

Verhalten

(en: behavior)

Angabe, wie das Objekt auf bestimmte Ereignisse reagiert

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

3.1.5.13

Klasse

(en: class)

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

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

3.1.5.14

Klassenattribute

(en: class attributes)

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

3.1.5.15

Klassencode

(en: class code)

eindeutige Kennung, die jeder Objektklasse zugewiesen ist

3.1.5.16

klassenspezifischer Dienst

(en: class specific service)

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

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

3.1.5.17

Client

(en: client)

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

3.1.5.18

Übertragungspfad

(en: conveyance path)

unidirektionaler Strom von APDUs über eine Anwendungsbeziehung

3.1.5.19

zyklisch

(en: cyclic)

Benennung zur Beschreibung von Ereignissen, die sich regelmäßig wiederholen

3.1.5.20

zugeordnete AR

(en: dedicated AR)

AR, die direkt vom FAL-Benutzer verwendet wird

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

3.1.5.21

Gerät

(en: device)

physikalische Hardware-Verbindung zum Verbindungszweig

ANMERKUNG Ein Gerät kann mehr als einen Knoten umfassen.

3.1.5.22

Geräteprofil

(en: device profile)

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

3.1.5.23

dynamische AR

(en: **dynamic AR**)

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

3.1.5.24

Endpunkt

(en: **endpoint**)

eine der Kommunikationseinheiten, die zu einer Verbindung gehören

3.1.5.25

Fehler

(en: **error**)

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

3.1.5.26

Fehlerklasse

(en: **error class**)

allgemeine Gruppierung für Fehlerdefinitionen

ANMERKUNG Innerhalb einer Fehlerklasse sind Fehlercodes für spezifische Fehler festgelegt.

3.1.5.27

Fehlercode

(en: **error code**)

Kennzeichnung eines bestimmten Fehlertyps innerhalb einer Fehlerklasse

3.1.5.28

FAL-Subnetz

(en: **FAL subnet**)

Netzwerke, die aus einem oder mehreren Segmenten auf Daten-Sicherungsschicht bestehen

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

3.1.5.29

logisches Gerät

(en: **logical device**)

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

3.1.5.30

Verwaltungsinformationen

(en: **management information**)

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

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

3.1.5.31

Netzwerk

(en: **network**)

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

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

3.1.5.32**Peer****(en: peer)**

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

3.1.5.33**vordefinierter AR-Endpunkt****(en: pre-defined AR endpoint)**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.1.6 Spezielle Begriffe zum Client/Server

3.1.6.1

Coils, diskrete Ausgänge

(en: coils, discrete outputs)

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

3.1.6.2

Discrete Input, diskreter Eingang

(en: discrete, discrete input)

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

3.1.6.3

Discrete Inputs, diskrete Eingänge

(en: discrete inputs, discretes)

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

3.1.6.4

Coil, diskreter Ausgang

(en: coil, discrete output)

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

3.1.6.5

gekapselte Schnittstelle

(en: encapsulated interface)

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

3.1.6.6

Ausnahme

(en: exception)

Codierung, die zur Signalisierung eines Fehlers der Dienstanforderung verwendet wird

3.1.6.7

Ausnahmecode

(en: exception code)

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

3.1.6.8

Datei

(en: file)

Anwendungsprozessobjekt; Gliederung von Einträgen, gekennzeichnet durch eine vorzeichenlose Zahl

3.1.6.9

Funktionscode

(en: function code)

Codierung eines Dienstes, der von einem Server angefordert wird

3.1.6.10**Holding Register, Ausgaberegister****(en: holding register, output register)**

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

3.1.6.11**Holding Registers, Ausgaberegister****(en: holding registers, output registers)**

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

3.1.6.12**Input Register****(en: input register)**

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

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

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

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

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

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

umgangssprachliche Benennung für Register

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

umgangssprachliche Benennung für Registertyp

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

Spezialisierung eines Funktionscodes

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

Kennung von logischen Geräten

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

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

3.1.7 Spezielle Begriffe zum Veröffentlichen/Abonnieren

3.1.7.1

Netzwerkobjekt

(en: network object)

Publish/Subscribe-Anwendung, Reader oder Writer

3.1.7.2

GUID

(en: GUID)

global eindeutige Kennung von Netzwerkobjekten zur eindeutigen Referenzierung eines Objektes innerhalb des Netzwerks

3.1.7.3

zusammengesetzter Zustand

(en: composite state)

Attribute einer Menge von Netzwerkobjekten

3.1.7.4

Composite State Transfer (CST)

(en: composite state transfer)

Interaktionen zwischen CST-Writers und interessierten CST-Readers mit dem Ziel, den CST-Readers die Rekonstruktion des zusammengesetzten Zustands des kommunizierenden CST-Writer zu erlauben, ohne die gesamte Historie, die zum aktuellen zusammengesetzten Zustand dieses CST-Writer geführt hat, zu übertragen

3.1.7.5

Reader

(en: reader)

Subscriber oder ein CST-Reader

3.1.7.6

Writer

(en: writer)

Publisher oder ein CST-Writer

3.1.7.7

CST-Reader

(en: CSTReader)

auf Metainformationen spezialisierter Subscriber

3.1.7.8

CST-Writer

(en: CSTWriter)

auf Metainformationen spezialisierter Publisher

3.1.7.9

Kommunikationsakteur

(en: communication actor)

Reader oder Writer

3.1.7.10

Domänenteilnehmer

(en: domain participant)

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

ANMERKUNG Diese Terminologie wurde eingeführt, um den übermäßigen Gebrauch der Benennung „Anwendung“ zu vermeiden. Gleichzeitig nimmt die Benennung „Domäne“ einen Platz innerhalb von „Publish/Subscribe“ ein. Die Erweiterungsmöglichkeit des Typs lässt das Konzept der „Domänen“ oder unabhängigen Kommunikationsebenen zu, wodurch eine effektive Isolierung des Anwendungsaustausches innerhalb von Domänen zugelassen wird. Während der

DDS von OMG, wie in „Data Distribution Service for Real-Time Systems Specification, Version 1.1, December 2005“, diese Erweiterung verwendet, wird diese Möglichkeit in der vorliegenden Spezifikation nicht weiter untersucht, sondern eine einzelne Domäne betrachtet.

3.1.7.11

Manager

(en: manager)

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

3.1.7.12

verwalteter Teilnehmer

(en: managed participant)

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

3.1.7.13

veröffentlichender Manager

(en: publishing manager)

Rolle eines AR-Endpunktes, in der dieser eine oder mehrere bestätigte Dienstanforderungs-APDUs an einen Publisher ausgibt, um die Veröffentlichung eines angegebenen Objektes vom Publisher anzufordern. In dieser Norm sind zwei Typen von veröffentlichenden Managern festgelegt: veröffentlichende Pull-Manager und veröffentlichende Push-Manager, die jeweils einzeln definiert sind

3.1.7.14

Pull-Publisher

(en: pull publisher)

Typ von Publisher, der ein Objekt als Antwort auf eine von dem entsprechenden veröffentlichenden Pull-Manager empfangende Anforderung veröffentlicht

3.1.7.15

veröffentlichender Pull-Manager

(en: pull publishing manager)

Typ von veröffentlichendem Manager, der die Veröffentlichung eines bestimmten Objektes in einer entsprechenden Antwort-APDU anfordert

3.1.7.16

Push-Publisher

(en: push publisher)

Typ von Publisher, der ein Objekt in einer unbestätigten Dienstanforderungs-APDU veröffentlicht

3.1.7.17

veröffentlichender Push-Manager

(en: push publishing manager)

Typ von veröffentlichendem Manager, der die Veröffentlichung eines bestimmten Objektes mithilfe eines unbestätigten Dienstes anfordert

3.1.7.18

Pull-Subscriber

(en: pull subscriber)

Typ von Subscriber, der empfangene bestätigte Dienstantwort-APDUs als veröffentlichte Objektdaten erkennt

3.1.7.19

Push-Subscriber

(en: push subscriber)

Typ von Subscriber, der empfangene unbestätigte Dienstanforderungs-APDUs als veröffentlichte Objektdaten erkennt

3.1.7.20

Sequenznummer

(en: **sequence number**)

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

English version

**Industrial communication networks -
Fieldbus specifications -
Part 5-15: Application layer service definition -
Type 15 elements
(IEC 61158-5-15:2010)**

Réseaux de communication industriels -
Spécifications des bus de terrain -
Partie 5-15: Définition des services des
couches d'application -
Éléments de type 15
(CEI 61158-5-15:2010)

Industrielle Kommunikationsnetze -
Feldbusse -
Teil 5-15: Dienstfestlegungen des
Application Layer (Anwendungsschicht) -
Typ 15-Elemente
(IEC 61158-5-15:2010)

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

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

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

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

CENELEC

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

Management Centre: Avenue Marnix 17, B - 1000 Brussels

Foreword

The text of document 65C/606/FDIS, future edition 2 of IEC 61158-5-15, prepared by SC 65C, "Industrial networks", of IEC/TC 65, "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61158-5-15:2012.

The following dates are fixed:

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

This document supersedes EN 61158-5-15:2008.

EN 61158-5-15:2012 includes the following significant technical change with respect to EN 61158-5-15:2008:

— editorial corrections.

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

Endorsement notice

The text of the International Standard IEC 61158-5-15:2010 was approved by CENELEC as a European Standard without any modification.

CONTENTS

INTRODUCTION	6
1 Scope	7
1.1 Overview	7
1.2 Specifications	8
1.3 Conformance	8
1.4 Type overview	8
2 Normative references	9
3 Terms and definitions, abbreviations, symbols and conventions	9
3.1 Terms and definitions	9
3.2 Abbreviations and symbols	17
3.3 Conventions	18
4 Concepts	21
4.1 Common concepts	21
4.2 Client/server specific concepts	21
4.3 Publish/subscribe specific concepts	30
5 Data type ASE	38
5.1 General	38
5.2 Formal definition of data type objects	38
5.3 FAL defined data types	38
5.4 Data type ASE service specification	51
6 Client/server communication model specification	51
6.1 ASEs	51
6.2 ARs	109
6.3 Summary of FAL classes	111
6.4 Permitted FAL services by AREP role	111
7 Publish/subscribe communication model specification	113
7.1 ASEs	113
7.2 ARs	131
7.3 Summary of FAL classes	132
7.4 Permitted FAL services by AREP role and sub-role	133
Bibliography	134
Annex ZA (normative) Normative references to international publications with their corresponding European publications	135
Figure 1 – Client/server stacks	21
Figure 2 – Client/server communication on different buses or networks	22
Figure 3 – Client/server APOs services conveyed by the FAL	22
Figure 4 – Interpretation as distinct tables	24
Figure 5 – Interpretation as overlapping tables	24
Figure 6 – APO and real objects, non obvious possible interpretation	25
Figure 7 – ASE service conveyance	26
Figure 8 – Client/server confirmed interaction	28
Figure 9 – Client/server AR confirmed service primitives (positive case)	28
Figure 10 – Client/server AR confirmed service primitives (negative case)	29

Figure 11 – Client/server unconfirmed interaction	29
Figure 12 – Client/server AR unconfirmed service primitives	30
Figure 13 – Publish/subscribe communications stacks	31
Figure 14 – Publish/subscribe data-centric exchanges between decoupled network objects	32
Figure 15 – Publish/subscribe APOs services conveyed by the FAL	32
Figure 16 – Examples of publish/subscribe configurable behaviors via QoS	34
Figure 17 – Pull model interactions	35
Figure 18 – Push model interactions	36
Figure 19 – Publish/subscribe model interactions	37
Figure 20 – Status bit sequence numbering	41
Figure 21 – ObjectId	45
Figure 22 – Bitmap	49
Figure 23 – ParameterSequence	51
Figure 24 – FAL ASEs	52
Figure 25 – Client/server encapsulated interface mechanism	98
Figure 26 – Publish/subscribe class derivations and relationships	113
Figure 27 – FAL ASEs and classes	114
Figure 28 – Publish/subscribe service request composition	123
Table 1 – Common client/server APOs	23
Table 2 – Class identification	46
Table 3 – Assigned vendor IDs	47
Table 4 – Bitmap “1234/12:00110”	49
Table 5 – Filter service parameters	54
Table 6 – Read discretes service parameters	57
Table 7 – Read coils service parameters	60
Table 8 – Write single coil service parameters	62
Table 9 – Write multiple coils service parameters	63
Table 10 – Broadcast write single coil service parameters	65
Table 11 – Broadcast write multiple coils service parameters	65
Table 12 – Read input registers service parameters	68
Table 13 – Read holding registers service parameters	73
Table 14 – Write single holding register service parameters	74
Table 15 – Write multiple holding registers service parameters	76
Table 16 – Mask write holding register service parameters	78
Table 17 – Read/write holding registers service parameters	80
Table 18 – Read FIFO service parameters	82
Table 19 – Broadcast write single holding register service parameters	83
Table 20 – Broadcast write multiple holding registers service parameters	84
Table 21 – Read file service parameters	91
Table 22 – Write file service parameters	95
Table 23 – Device identification categories	100
Table 24 – Read device ID code	101

Table 25 – Conformity level.....	102
Table 26 – Requested vs. returned known objects	103
Table 27 – Read device identification service parameters.....	105
Table 28 – FAL class summary	111
Table 29 – Services by AREP role	112
Table 30 – Issue service parameters	116
Table 31 – Heartbeat service parameters	117
Table 32 – VAR service parameters	118
Table 33 – VAR service parameters	120
Table 34 – ACK service parameters	122
Table 35 – Header service parameters	125
Table 36 – INFO_DST service parameters	126
Table 37 – INFO_REPLY service parameters	127
Table 38 – INFO_SRC service parameters.....	129
Table 39 – INFO_TS service parameters.....	130
Table 40 – PAD service parameters	131
Table 41 – FAL class summary	133
Table 42 – Services by AREP role and sub-role	133

INTRODUCTION

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-15: Application layer service definition – Type 15 elements

1 Scope

1.1 Overview

In network communications, as in many fields of engineering, it is a fact that “one size does not fit all.” Engineering design is about making the right set of trade-offs, and these trade-offs must balance conflicting requirements such as simplicity, generality, ease of use, richness of features, performance, memory size and usage, scalability, determinism, and robustness. These trade-offs must be made in light of the types of information flow (e.g. periodic, one-to-many, request-reply, events), and the constraints imposed by the application and execution platforms.

The Type 15 fieldbus provides two major communication mechanisms that complement each others to satisfy communication requirements in the field of automation: the Client/Server and the Publish/Subscribe paradigms. They can be used concurrently on the same device.

Type 15 Client/Server operates in a Client/Server relationship. Its application layer service definitions and protocol specifications are independent of the underlying layers, and have been implemented on a variety of stacks and communication media, including EIA/TIA-232, EIA/TIA-422, EIA/TIA-425, HDLC (ISO 13239), fiber, TCP/IP, Wireless LANs and Radios.

Type 15 Publish/Subscribe operates in a Publish/Subscribe relationship. Its application layer service definitions and protocol specifications are independent of the underlying layers and can be configured to provide reliable behavior and support determinism. The most common stack is UDP/IP.

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 part of IEC 61158 provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 15 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This part of IEC 61158 defines in an abstract way the externally visible service provided by the Type 15 fieldbus application layer in terms of

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

The purpose of this part of IEC 61158 is to define the services provided to

- a) the FAL user at the boundary between the user and the Application Layer of the Fieldbus Reference Model, and

- b) Systems Management at the boundary between the Application Layer and Systems Management of the Fieldbus Reference Model.

This part of IEC 61158 specifies the structure and services of the Type 15 IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI Application Layer Structure (ISO/IEC 9545).

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

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

1.2 Specifications

The principal objective of this part of IEC 61158 is to specify the characteristics of conceptual application layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of application layer protocols for time-critical communications.

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

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

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

1.3 Conformance

This part of IEC 61158 does not specify individual implementations or products, nor do they constrain the implementations of application layer entities within industrial automation systems.

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill the Type 15 application layer services as defined in this part of IEC 61158.

1.4 Type overview

In network communications, as in many fields of engineering, it is a fact that “one size does not fit all.” Engineering design is about making the right set of trade-offs, and these trade-offs must balance conflicting requirements such as simplicity, generality, ease of use, richness of features, performance, memory size and usage, scalability, determinism, and robustness. These trade-offs must be made in light of the types of information flow (e.g. periodic, one-to-many, request-reply, events), and the constraints imposed by the application and execution platforms.

The Type 15 fieldbus provides two major communication mechanisms that complement each others to satisfy communication requirements in the field of automation: the Client/Server and the Publish/Subscribe paradigms. They can be used concurrently on the same device.

Type 15 Client/Server operates in a Client/Server relationship. Its application layer service definitions and protocol specifications are independent of the underlying layers, and have been implemented on a variety of stacks and communication media, including EIA/TIA-232, EIA/TIA-422, EIA/TIA-425, HDLC (ISO 13239), fiber, TCP/IP, Wireless LANs and Radios.

Type 15 Publish/Subscribe operates in a Publish/Subscribe relationship. Its application layer service definitions and protocol specifications are independent of the underlying layers and can be configured to provide reliable behavior and support determinism. The most common stack is UDP/IP.

2 Normative references

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

IEC/TR 61158-1:2010¹, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-6-15:2010¹, *Industrial communication networks – Fieldbus specifications – Part 6-15: Application layer protocol specification – Type 15 elements*

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

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

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

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

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

3 Terms and definitions, abbreviations, symbols and conventions

3.1 Terms and definitions

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

3.1.1 ISO/IEC 7498-1 terms

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

¹ To be published.

- i) transfer syntax

3.1.2 ISO/IEC 8822 terms

- a) abstract syntax
- b) presentation context

3.1.3 ISO/IEC 9545 terms

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

3.1.4 ISO/IEC 8824 terms

- a) object identifier
- b) type

3.1.5 IEC/TR 61158-1 terms

The following IEC/TR 61158-1 terms apply.

3.1.5.1

application

function or data structure for which data is consumed or produced

3.1.5.2

application layer interoperability

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

3.1.5.3

application object

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

NOTE Multiple types of application object classes may be defined.

3.1.5.4

application process

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

3.1.5.5

application process identifier

distinguishes multiple application processes used in a device

3.1.5.6

application process object

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

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

3.1.5.7

application process object class

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

3.1.5.8

application relationship

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

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

3.1.5.9

application relationship endpoint

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

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

3.1.5.10

application service element

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

3.1.5.11

attribute

description of an externally visible characteristic or feature of an object

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

3.1.5.12

behavior

indication of how the object responds to particular events

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

3.1.5.13

class

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

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

3.1.5.14

class attributes

attribute that is shared by all objects within the same class

3.1.5.15

class code

unique identifier assigned to each object class

3.1.5.16

class specific service

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

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

3.1.5.17

client

(a) object which uses the services of another (server) object to perform a task

(b) initiator of a message to which a server reacts, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

3.1.5.18

conveyance path

unidirectional flow of APDUs across an application relationship

3.1.5.19

cyclic

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

3.1.5.20

dedicated AR

AR used directly by the FAL user

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

3.1.5.21

device

physical hardware connection to the link

NOTE A device may contain more than one node.

3.1.5.22

device profile

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

3.1.5.23

dynamic AR

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

3.1.5.24

endpoint

one of the communicating entities involved in a connection

3.1.5.25

error

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

3.1.5.26

error class

general grouping for error definitions

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

3.1.5.27

error code

identification of a specific type of error within an error class

3.1.5.28

FAL subnet

networks composed of one or more data link segments

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

3.1.5.29

logical device

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

3.1.5.30
management information

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

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

3.1.5.31
network

series of nodes connected by some type of communication medium

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

3.1.5.32
peer

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

3.1.5.33
pre-defined AR endpoint

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

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

3.1.5.34
pre-established AR endpoint

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

3.1.5.35
publisher

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

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

3.1.5.36
server

a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

b) object which provides services to another (client) object

3.1.5.37
service

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

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

3.1.5.38
subscriber

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

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

3.1.6 Specific definitions for client/server

3.1.6.1

coils, discrete outputs

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

3.1.6.2

discrete, discrete input

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

3.1.6.3

discrete inputs, discretes

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

3.1.6.4

coil, discrete output

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

3.1.6.5

encapsulated interface

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

3.1.6.6

exception

encoding used to signal a service request failure

3.1.6.7

exception code

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

3.1.6.8

file

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

3.1.6.9

function code

encoding of a service requested to a server

3.1.6.10

holding register, output register

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

3.1.6.11

holding registers, output registers

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

3.1.6.12

input register

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

3.1.6.13

input registers

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

3.1.6.14

record

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

3.1.6.15

reference

denigrated term for register

3.1.6.16

reference type

denigrated term for register type

3.1.6.17

sub-code

specialization of a function code

3.1.6.18

unit ID

logical device identifier

3.1.6.19

MEI type

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

3.1.7 Specific definitions for publish/subscribe

3.1.7.1

network object

publish/subscribe application, reader, or writer

3.1.7.2

GUID

globally unique network object identifier, used to uniquely reference an object within the network

3.1.7.3

composite state

attributes of a set of network objects

3.1.7.4

composite state transfer

Interactions between CSTWriters and interested CSTReaders with the goal to allow the CSTReaders to reconstruct the composite state of the communicating CSTWriter, without transferring the entire history that led to the current composite state of that CSTWriter.

3.1.7.5

reader

subscriber or a CSTReader

3.1.7.6

writer

publisher or a CSTWriter

3.1.7.7

CSTReader

meta-information-specialized subscriber

3.1.7.8

CSTWriter

meta-information-specialized publisher

3.1.7.9

communication actor

reader or writer

3.1.7.10

domain participant

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

NOTE This terminology is adopted to avoid the overuse of the term "application". At the same time, the term "domain" has a place within publish/subscribe. The Type extensibility allows for the concept of "domains", or independent communication planes, effectively permitting isolation of application exchanges within domains. While OMG DDS as in "Data Distribution Service for Real-Time Systems Specification, Version 1.1, December 2005" uses this extension, the feature will not be examined further in this specification, which will consider a single domain.

3.1.7.11

manager

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

3.1.7.12

managed participant

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

3.1.7.13

publishing manager

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

3.1.7.14

pull publisher

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

3.1.7.15

pull publishing manager

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

3.1.7.16

push publisher

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

3.1.7.17

push publishing manager

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

3.1.7.18

pull subscriber

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

3.1.7.19

push subscriber

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

3.1.7.20

sequence number

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

3.2 Abbreviations and symbols

3.2.1 Common abbreviations and symbols

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

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

3.2.2 Abbreviations and symbols for client/server

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

3.2.3 Abbreviations and symbols for publish/subscribe

CS	Composite State
CST	Composite State Transfer
DCPS	Data-Centric Publish-Subscribe
DDS	Data Distribution Service
DLRL	Data Local Reconstruction Layer
GUID	Globally Unique Id
OMG	Object Management Group
P/S	Publish/Subscribe

3.3 Conventions

3.3.1 Overview

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

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

3.3.2 General conventions

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

3.3.3 Conventions for class definitions

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

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

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

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

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

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

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

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

3.3.4 Conventions for service definitions

3.3.4.1 General

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

3.3.4.2 Service parameters

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

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

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

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

M parameter is mandatory for the primitive

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

C parameter is conditional upon other parameters or upon the environment of the service user.

— (blank) parameter is never present.

S parameter is a selected item.

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

a) a parameter-specific constraint:

"(=)" indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.

b) an indication that some note applies to the entry:

"(n)" indicates that the following note "n" contains additional information pertaining to the parameter and its use.

3.3.4.3 Service procedures

The procedures are defined in terms of

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

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

4 Concepts

4.1 Common concepts

All of IEC/TR 61158-1:2010, Clause 9 is incorporated by reference, except as specifically overridden in 4.2 and 4.3.

4.2 Client/server specific concepts

4.2.1 Overview

Client/Server (C/S) describes a Client/Server communication mechanism between devices connected on different types of buses or networks. The client/server AL is the same irrelevant from the underlying layers. Some client/server stack configurations are illustrated in Figure 1.

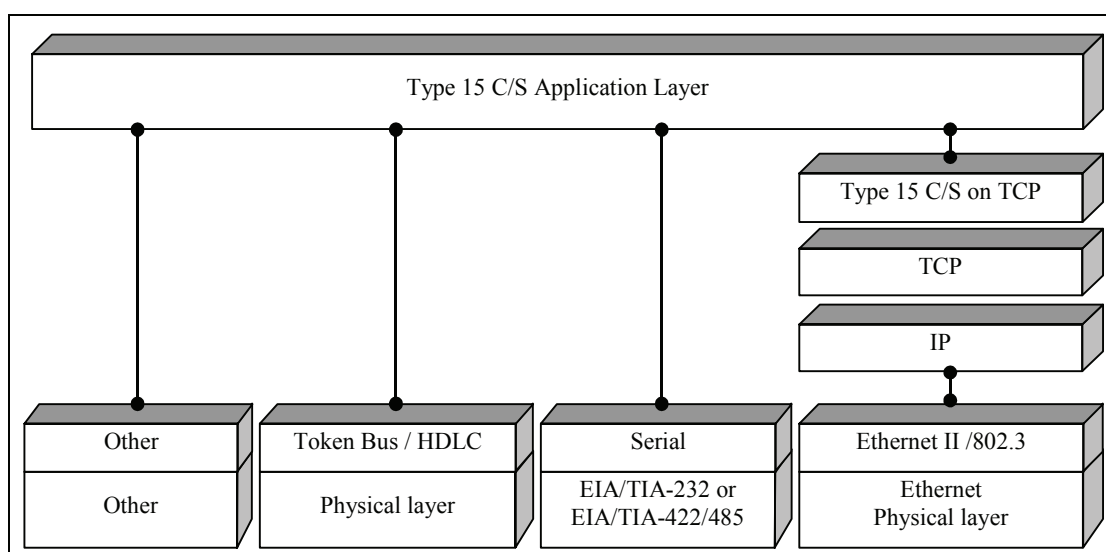


Figure 1 – Client/server stacks

The communication between devices on different types of buses or networks is made possible by the use of gateways, as illustrated in Figure 2.

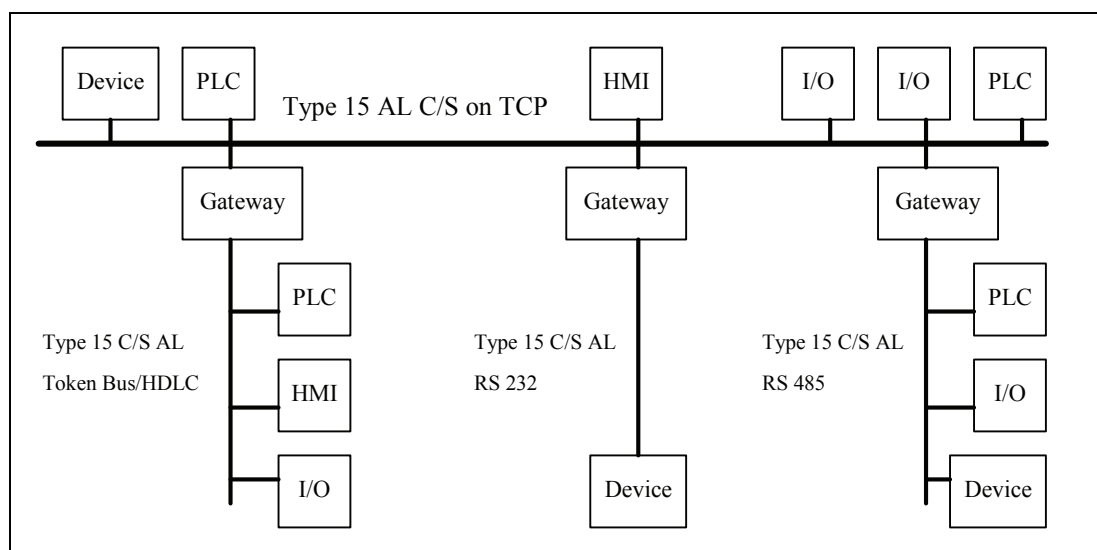


Figure 2 – Client/server communication on different buses or networks

4.2.2 Client/server APOs

4.2.2.1 General

Client/server provides application layer services specified by function codes, acting on application process objects (APOs).

An application process object is a network representation of a specific aspect of an AP. Each APO represents a specific set of information and processing capabilities of an AP that are accessible through services of the FAL. APOs are used to represent these capabilities to other APs in a fieldbus system.

From the perspective of the FAL, an APO is modeled as a network accessible object contained within an AP or within another APO (APOs may contain other APOs). APOs provide the network definition for objects contained within an AP that are remotely accessible. The definition of an APO includes an identification of the FAL services that can be used by remote APs for remote access. The FAL services, as shown in Figure 3, are provided by the FAL communications entity of the AP, known as the FAL Applications Entity (FAL AE).

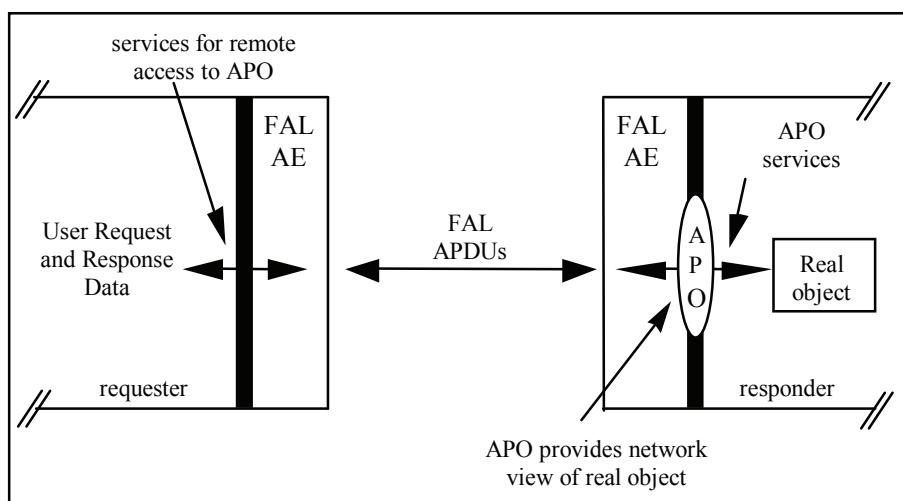


Figure 3 – Client/server APOs services conveyed by the FAL

In Figure 3, remote APs acting as clients may access the real object by sending requests through the APO that represents the real object. Local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

For all APOs, the semantics is the one allowed by the associated FAL APO ASE.

4.2.2.2 Common APOs

Client/server bases its model on APOs that have distinguishing characteristics. Some of them, namely discretes, coils, input registers and holding registers, are often interpreted as memory tables. While this is a simple and convenient model, it is just one of many, indeed they do not even need to be seen as memory objects. It is up to the application to devise the most appropriate relation of APOs with real objects, and the subject is out of the scope of this document. Table 1 summarizes the characteristics of these four APOs.

Table 1 – Common client/server APOs

Common APOs	Object type	Comment
discrete	single bit	Read-only, its value can be provided by an I/O system. This APO is useful in the modeling of binary valued real objects that are manipulated by the server application and are supposed to be only observed by the client user. The integrity of the above contract is under control of the server AP, which can confine the exposure of the real objects to discretes
coil	single bit	Read-write, its value can be altered by a client application program
input register	16 bit word	Read-only, its value can be provided by an I/O system. This APO is useful in the modeling of analog valued real objects that are manipulated by the server application and are supposed to be only observed by the client user. The integrity of the above contract is under control of the server AP, which can confine the exposure of the real objects to input registers
holding register	16 bit word	Read-write, its value can be altered by a client application program

The distinctions between inputs and outputs, and between bit-addressable and word-addressable data items, do not imply any application behavior. It is perfectly acceptable, and customary, to regard all four tables as overlaying one another, if this is the most natural interpretation on the target machine in question.

The following Figure 4 and Figure 5, informative, give some common but by no means exhaustive interpretations, respectively as distinct memory tables and overlapping memory tables.

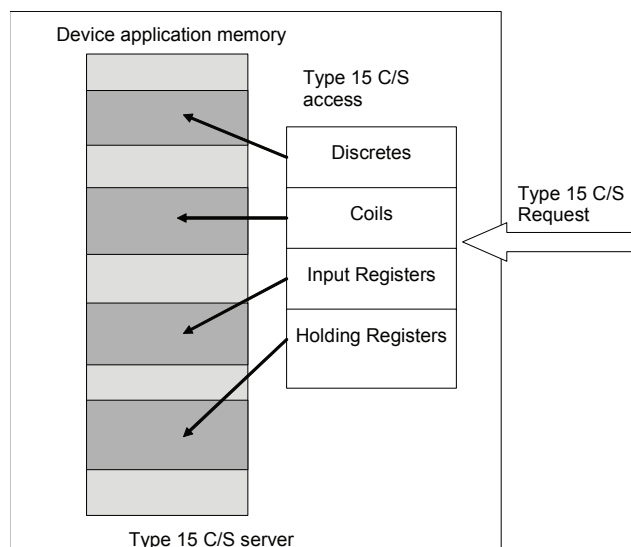


Figure 4 – Interpretation as distinct tables

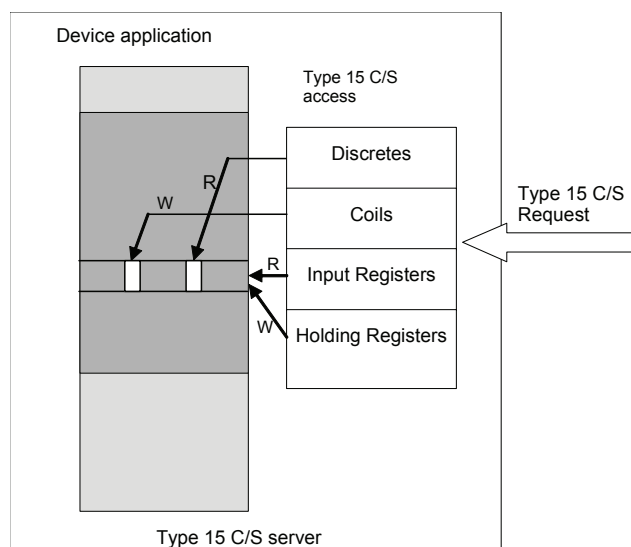


Figure 5 – Interpretation as overlapping tables

To emphasize what is not implied, it is worth mentioning yet another not obvious interpretation (informative) often found in the field: different requests performed on the same APO do not necessarily entail requests to the same real object. This is illustrated in Figure 6, informative.

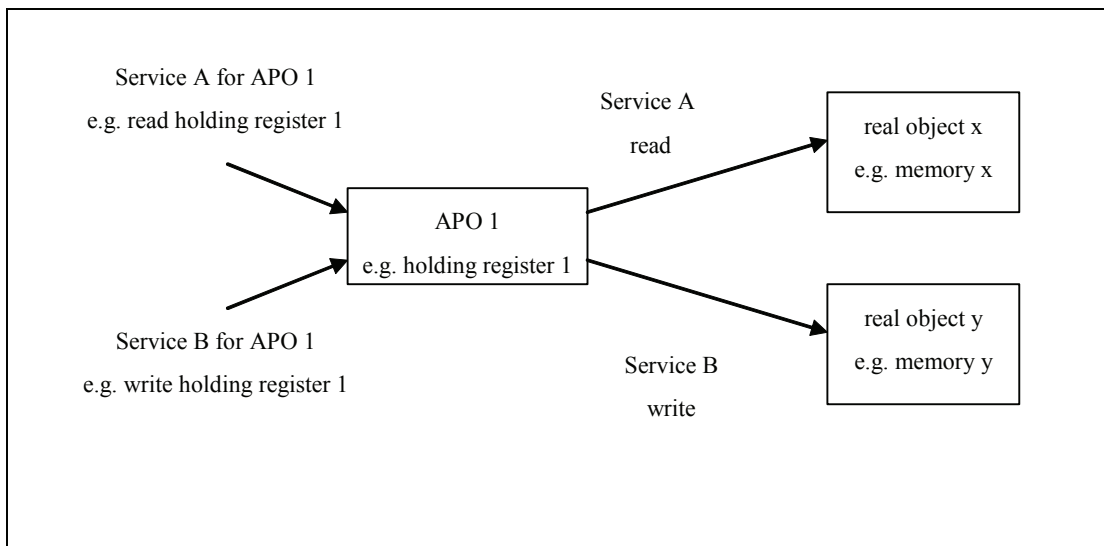


Figure 6 – APO and real objects, non obvious possible interpretation

Discretes, coils, input registers and holding registers are often collectively called data references or data items. For each of the above mentioned tables, the protocol allows individual selection of 65 536 data items, and the operations of read or write of those items are designed to span, from an application user (client user) point of view, multiple consecutive data items up to a data size limit which is dependent on the service transaction function code.

The possible association of client/server data references (bits, registers) and actual physical storage within devices is a local issue.

The client/server logical data reference addresses, used in client/server service function codes, are unsigned integers starting at zero.

One potential source of confusion is the relationship between the data reference addresses used in client/server service function codes, and the discrete, coil or register 'user' reference addresses, as seen e.g. in a PLC Ladder Logic program. For historical reasons user reference addresses were expressed as decimal numbers with a starting offset of 1, however client/server uses the more natural software convention starting at 0. Due to this, for example, a client/server message requesting to read a register via logical data reference at offset 0 would return what the application programmer considers register 1. There is an exception, the addressing of records via registers using client/server service function codes Read File Record and Write File Record, where record 0, a 'user' reference, is addressed specifying 0 as its first register. Details are given in the mentioned services specifications.

4.2.2.3 Record and file APOs

The record APO, from an application user (client user) point of view, is a set of contiguous registers of a specified type, characterized by the address of the first register and by the quantity of registers; in the context of this definition, the registers involved have also been called references.

The file APO is an organization of records, characterized by an unsigned number.

4.2.2.4 Interface APO

The encapsulated interface is a mechanism encapsulating a service for an interface, which is an application process object characterized by an MEI type. The MEI type, specified as an octet value, is used to dispatch the service to the appropriate interface.

While it is a simple mechanism, since it defers all the semantics to the interface, it allows maximum flexibility in what it can represent and convey.

4.2.3 Client/server function codes

Function codes are encodings of services requested to a server. These encodings are partitioned in three categories:

Publicly assigned function codes

These function codes are either assigned to a standard service or reserved for future assignment.

User definable function codes

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

Reserved function codes

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

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

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

4.2.4 Client/server communication model overview

4.2.4.1 General

Figure 7 is a representation of the end-to-end communication model.

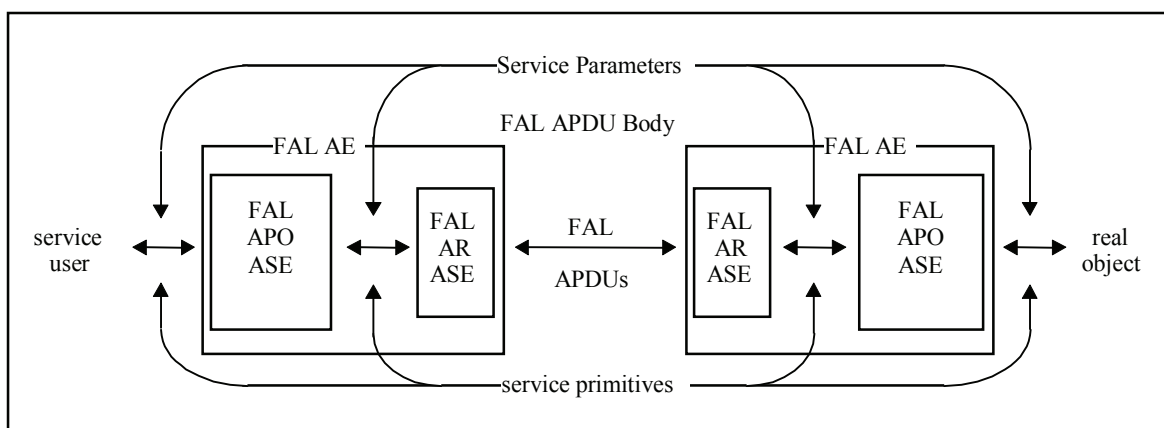


Figure 7 – ASE service conveyance

Client/server uses a Client/Server AR with confirmed interaction, and optionally when broadcast is supported, a Client/Server AR with unconfirmed interaction.

4.2.4.2 Client/server device addressing and connection

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

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

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

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

NOTE In general the Unit ID is only required for logical devices having the role of servers. Often logical devices can have either role or multiple roles, via configuration, and their Unit ID is not used when they only perform in the client role. Depending on the underlying

layers some devices can have concurrently a client and a server role on the same access point, this is the case for client/server on Token Bus/HDLC, outside the scope of this standard.

4.2.4.3 Client/server cardinality, connections and transaction objects

A client/server device may have many clients and many servers, depending on the device and on the underlying layers.

The clients and servers are associated by a connection, which in the scope of this standard is equivalent to an AR.

Clients carry a service on a connection with the help of a transaction object, managed (created and destroyed) by the client. The transaction object mechanism is exposed at the application layer due to the client/server possibility of having more than one outstanding request at a time, with the consequent need to properly associate requests and confirmations. It also controls the maximum number of such requests, which could be 1. The capabilities of a client/server application layer client depend on lower layers and on the particular implementation; these factors are captured in the configuration of the transaction object mechanism allowing programmatic adaptation.

The transaction object is instantiated for the duration of a service invocation. Every transaction object needs to be uniquely identifiable within a connection.

There can be many servers on the same connection. There can be many clients on the same connection as well, but when this is the case then only one client on that connection can be active at a time. For confirmed services, a client is active from the time it sends a request to the time it receives the corresponding confirmation, and this time is contained within the instantiation and subsequent destruction of the transaction object involved. For unconfirmed services, a client is active for an amount of time that is device and connection specific, and it shall itself wait for a previous activity time to elapse before engaging in another one. The enforcement of the single active client per connection at a time is outside the scope of this specification, as is the timing of activities for unconfirmed services.

A client can issue only one service invocation per transaction object. Depending on the device there may be only one connection, or a limited number of connections. Depending on the underlying layer and local settings, a client may not be allowed to instantiate more than a limited number of transaction objects at a time.

In case of connections where multiple clients per connection are allowed, those clients can only instantiate one transaction object at a time, obtaining the above mentioned activity status when they do so. Some connections with a single client may have the same restriction.

If a client can only instantiate one transaction object at a time then, when invoking a service, it is not necessary to dynamically create a transaction object and exchange it with lower layers, since the main reason for having a transaction object is to properly couple requests with confirmations, which is automatic for these clients. In these situations a single static transaction object effectively acts like a client token, which is either taken or available, and only of interest to the client.

In all cases, the establishment of a connection and the needed configuration for the transaction object instantiation are outside the scope of this specification. Both are expected to be properly established.

4.2.4.4 Confirmed interaction

Client/server is based on request/response messages, and the model is that of a client/server interaction. With this relationship, a client issues a request as a confirmed service to a server, and the confirmation of the response from the server concludes the transaction. This interaction is shown in Figure 8.

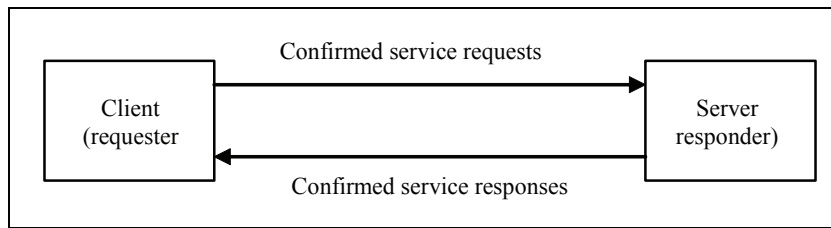


Figure 8 – Client/server confirmed interaction

This is accomplished using the Client/Server AR primitives, as shown in Figure 9 and Figure 10.

A client should start a transaction timer after issuing a confirmed service request. This timer should be set to an implementation dependent maximum time allowed for the confirmation, resetting it on receipt or reporting a service provider error on expiration.

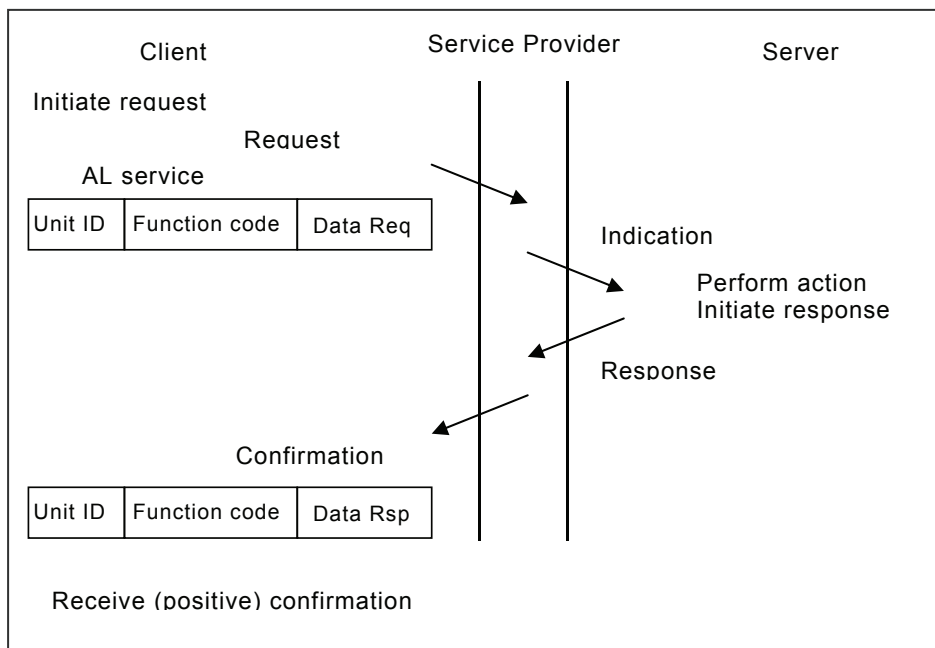


Figure 9 – Client/server AR confirmed service primitives (positive case)

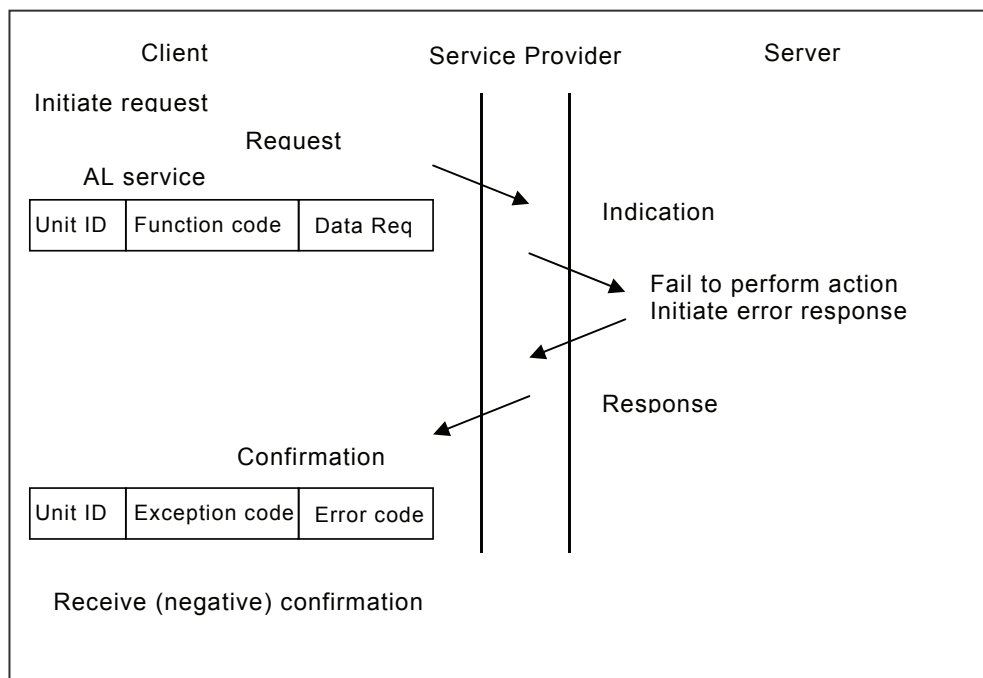


Figure 10 – Client/server AR confirmed service primitives (negative case)

4.2.4.5 Unconfirmed interaction

Optionally, clients and servers may support unconfirmed services in the form of client issued broadcast messages. This interaction is shown in Figure 11.

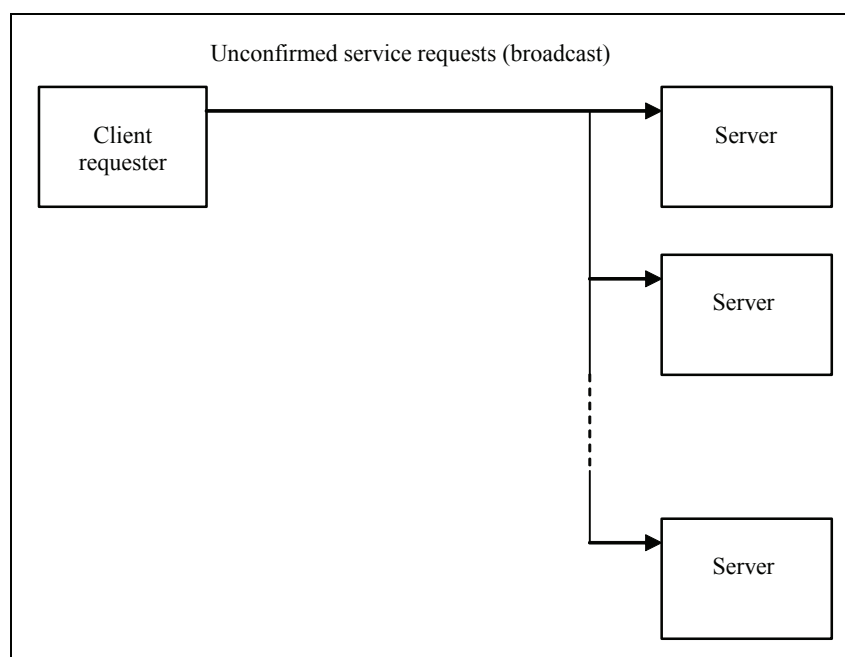


Figure 11 – Client/server unconfirmed interaction

This is accomplished using the Client/Server AR primitives, as shown in Figure 12.

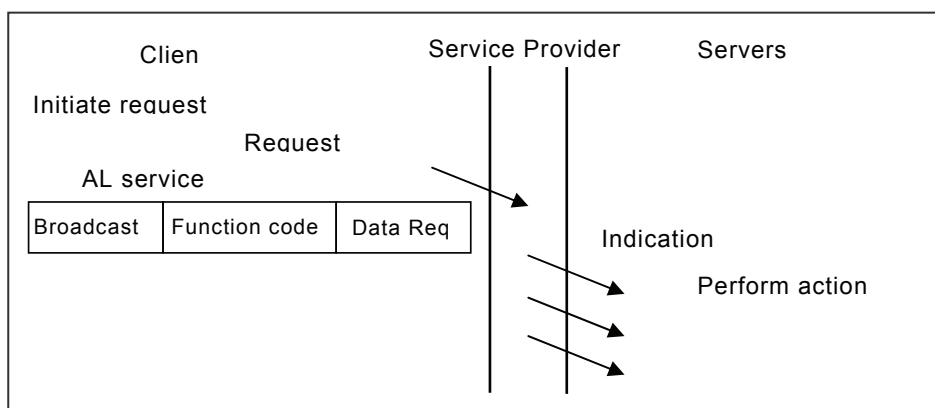


Figure 12 – Client/server AR unconfirmed service primitives

4.3 Publish/subscribe specific concepts

4.3.1 Overview

Publish/Subscribe (P/S) describes a Publish/Subscribe communication mechanism between networked devices. publish/subscribe offers a number of QoS characteristics, and it is designed with upward compatible extensibility features, so that, among other things, the QoS set can be easily augmented, using versioning or with vendor specific extensions. This specification will indicate the extension possibilities but it will not detail them, since outside the scope. It is important to realize that publish/subscribe is a wire protocol, it exposes many possibilities via configuration and extensions, and not all behaviors are documented in this specification.

NOTE A higher application services layer, dictating optimal usage and taking advantage of easy to accommodate 'designed-in' extensions for publish/subscribe, has been recently standardized by the OMG: "Data Distribution Service for Real-Time Systems Specification, Version 1.1, December 2005".

The publish/subscribe AL has minimal requirements on the underlying layers, and it could be deployed using memory offsets on shared memory or ports on UDP. publish/subscribe supports a wide variety of transports and transport QoS. The protocol is designed to be able to run on multicast and best-effort transports, such as UDP/IP, and requires only very simple services from the transport. In fact, it is sufficient that the transport offers a connectionless service capable of sending packets best-effort. That is, the transport need not guarantee each packet will reach its destination or that packets are delivered in-order. Where required, publish/subscribe implements reliability in the transfer of data and state above the transport interface. This does not preclude publish/subscribe from being implemented on top of a reliable transport. It simply makes it possible to support a wider range of transports.

The general requirements publish/subscribe asks to the underlying transport can be summarized as follows:

- the transport has a generalized notion of a unicast address (shall fit within 16 octets);
- the transport has a generalized notion of a port (shall fit within 4 octets), e.g. could be a UDP port, an offset in a shared memory segment, etc.;
- the transport can send a datagram (uninterpreted sequence of octets) to a specific address/port;
- the transport can receive a datagram at a specific address/port;
- the transport will drop messages if incomplete or corrupted during transfer (i.e. publish/subscribe assumes messages are complete and not corrupted);
- the transport provides a means to deduce the size of the received message.

A publish/subscribe stack configuration is illustrated in Figure 13.

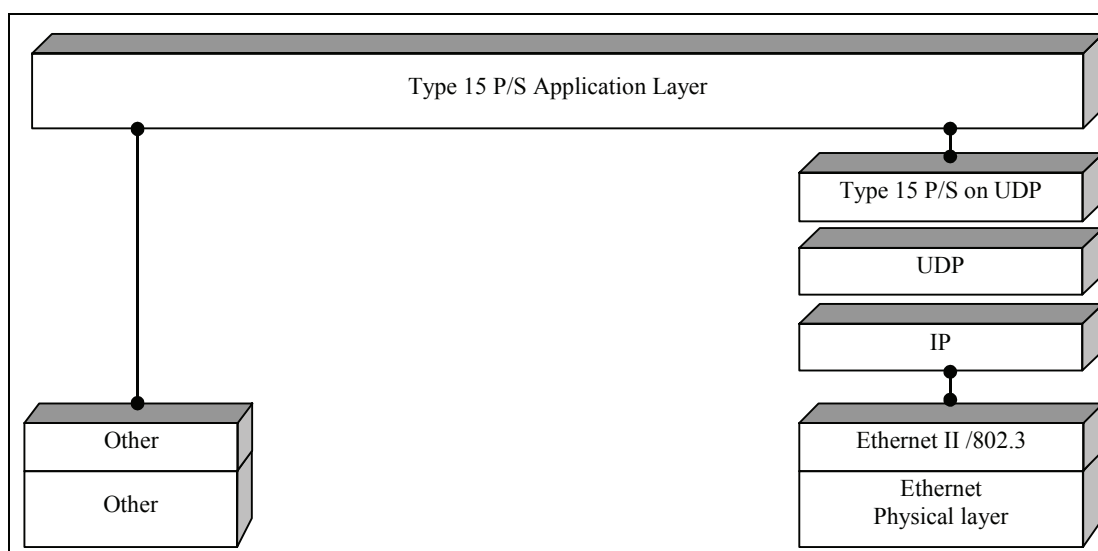


Figure 13 – Publish/subscribe communications stacks

If available, publish/subscribe can also take advantage of the multicast capabilities of the transport mechanism, where one message from a sender can reach multiple receivers.

Publish/subscribe is designed to promote determinism of the underlying communication mechanism; it provides an open trade-off between determinism and reliability. Among other features, determinism is monitored with the use of deadlines. Reliability is provided using queues, acknowledgements and retransmission. The reliability is window based, implemented natively, allowing full control and flexibility of the aforementioned trade-off.

4.3.2 Data-centric, match, decoupling and scalability

Publish/subscribe is organized around publishers and subscribers, with the additional characteristic of being data centric.

In this architecture, often called DCPS for data centric publish subscribe, the attributes of an exchange are on the data itself. In addition to the content proper, that may or may not be a discriminating factor, the operating traits of a publisher are reflected in the data it publishes, and the expectations of a subscriber are looked for in the data it subscribes to.

The above allows for a high degree of customization regarding data and its exchanges, and a match based on data, sometimes even down to parts of the content, with no direct knowledge between publishers and subscribers. This decoupling is the major reason for the scalability properties of this approach, it is easy to add and remove publishers and subscribers.

Figure 14 illustrates a scenario where applications exchange data via publishers and subscribers, unknown to each others.

The match is made by considering attributes of the data, like Topic name, Topic type, and other qualifying characteristics.

NOTE Publish/subscribe extensions allow for the concept of “domains”, effectively permitting isolation of application exchanges within domains. While OMG DDS as in “Data Distribution Service for Real-Time Systems Specification, Version 1.1, December 2005” uses this extension, the feature will not be examined further in this specification, which will consider a single domain.

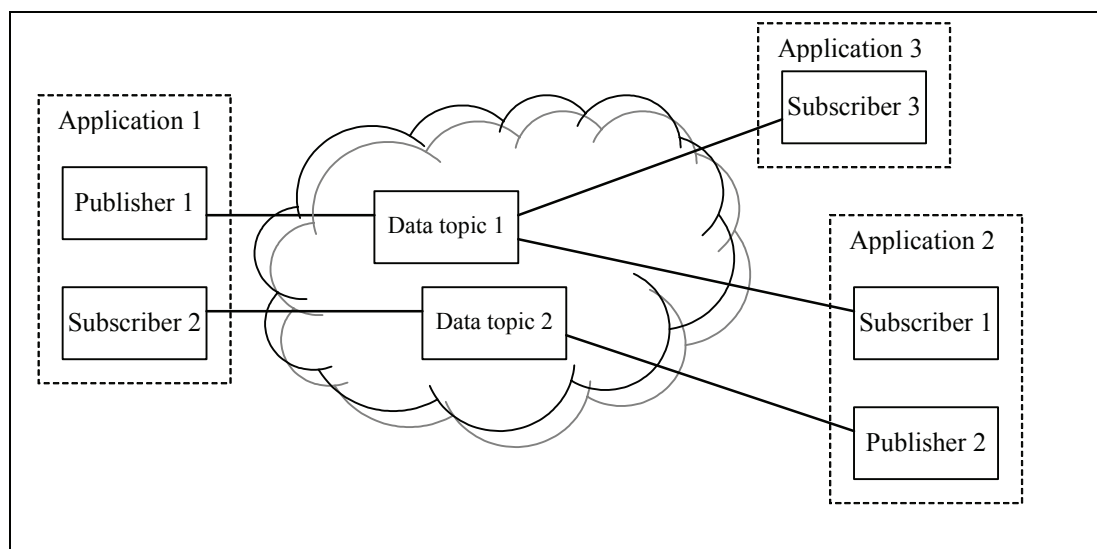


Figure 14 – Publish/subscribe data-centric exchanges between decoupled network objects

4.3.3 Publish/subscribe APOs

4.3.3.1 General

An application process object is a network representation of a specific aspect of an AP. Each APO represents a specific set of information and processing capabilities of an AP that are accessible through services of the FAL. APOs are used to represent these capabilities to other APs in a fieldbus system.

From the perspective of the FAL, an APO is modeled as a network accessible object contained within an AP or within another APO (APOs may contain other APOs). APOs provide the network definition for objects contained within an AP that are remotely accessible. The definition of an APO includes an identification of the FAL services that can be used by remote APs for remote access. The FAL services, as shown in Figure 15, are provided by the FAL communications entity of the AP, known as the FAL Applications Entity (FAL AE).

Publish/subscribe APOs are constructed dynamically, and are exchanged from publishers to subscribers.

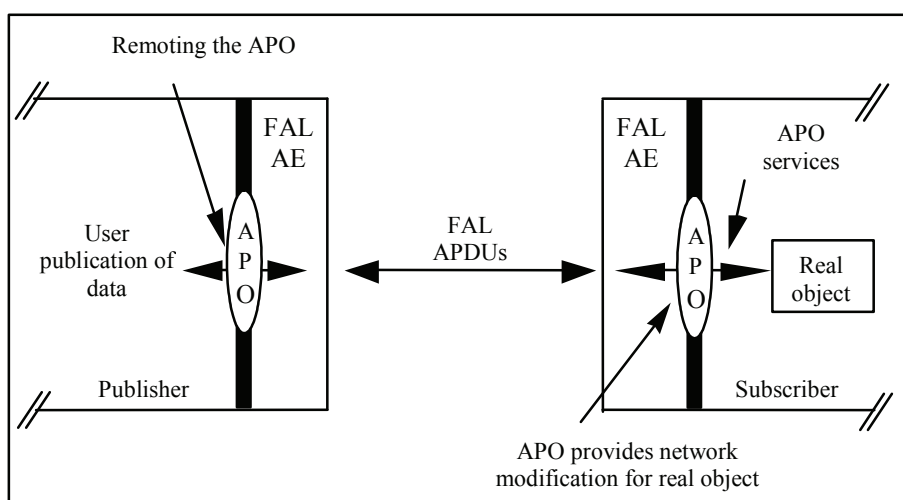


Figure 15 – Publish/subscribe APOs services conveyed by the FAL

In Figure 15, remote APs acting as publishers may modify the real object by publishing data that contributes changes to the remote APO. publish/subscribe remotes the APO itself, by publishing it. Once the APO is

acquired by the subscriber, local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

4.3.3.2 Data topic APO

The publish/subscribe APO is the data exchanged itself, the publication, here called topic to emphasize the fact that it is identifiable data.

The topic is characterized by the following:

- topic name;
- optional topic type;
- optional topic representational code.

The offered publication topic is looked at by an expected subscription topic, and a match is attempted. The matching is done on all three components, with the following considerations:

- the name components have to match, but this is not required to be a literal match, as it can be done via regular expression pattern matching;
- if the topic type of one of the matching sides is the empty string, then that component is a match, and the additional meaning is that no type checking against the type information carried by this component should be done; the type information is expected to be available by other means; if the topic type of both sides is not the empty string then they have to match;
- if the topic representational code of one of the matching sides is 0, then that component is a match, and the additional meaning is that there is no extra information available about the topic type and its marshalling; the type information is expected to be available by other means; if the topic representational code of both sides is not 0 then they have to match.

The offered publication topics and the expected subscription topics carry with themselves additional information about the data production and consumption, respectively, via optional parameters. This is the information that enables QoS, and it is available on a publisher, topic and subscriber basis. The kind and quantity of additional information is configurable; the following (Informative) is a commonly used set:

publication topic

- topic (name, type, representational code);
- strength;
- persistence.

The strength is the precedence of the issue sent by the Publication; the persistence indicates how long the issue is valid. Strength and persistence allow the receiver to arbitrate if issues are received from several matching publications.

subscription topic

- topic (name, type, representational code);
- minimum separation;
- deadline.

The minimum separation is the minimum time between two consecutive issues received by the Subscription. It defines the maximum rate at which the Subscription is prepared to receive issues. Publications sending to this Subscription should try to send issues so that they are spaced at least this far apart.

The above configuration enables the behaviors illustrated in Figure 16, informative.