

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Industrial communication networks – Fieldbus specifications –
Part 5-15: Application layer service definition – Type 15 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 5-15: Définition des services de la couche application – Éléments de
Type 15**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2007 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC

La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC

Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue

Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub

La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,...). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 14 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

Plus de 55 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Industrial communication networks – Fieldbus specifications –
Part 5-15: Application layer service definition – Type 15 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 5-15: Définition des services de la couche application – Éléments de
Type 15**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XF

ICS 25.040.40; 35.100.70

ISBN 978-2-8322-1557-9

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	5
INTRODUCTION.....	7
1 Scope.....	8
1.1 Overview.....	8
1.2 Specifications.....	9
1.3 Conformance.....	9
1.4 Type overview.....	9
2 Normative references.....	10
3 Terms and definitions, abbreviations, symbols and conventions.....	10
3.1 Terms and definitions.....	10
3.2 Abbreviations and symbols.....	18
3.3 Conventions.....	19
4 Concepts.....	23
4.1 Common concepts.....	23
4.2 Client/server specific concepts.....	23
4.3 Publish/subscribe specific concepts.....	31
5 Data type ASE.....	40
5.1 General.....	40
5.2 Formal definition of data type objects.....	40
5.3 FAL defined data types.....	40
5.4 Data type ASE service specification.....	53
6 Client/server communication model specification.....	54
6.1 ASEs.....	54
6.2 ARs.....	113
6.3 Summary of FAL classes.....	115
6.4 Permitted FAL services by AREP role.....	116
7 Publish/subscribe communication model specification.....	117
7.1 ASEs.....	117
7.2 ARs.....	136
7.3 Summary of FAL classes.....	138
7.4 Permitted FAL services by AREP role and sub-role.....	138
Bibliography.....	139
Figure 1 – Client/server stacks.....	23
Figure 2 – Client/server communication on different buses or networks.....	23
Figure 3 – Client/server APOs services conveyed by the FAL.....	24
Figure 4 – [INFORMATIVE] Interpretation as distinct tables.....	25
Figure 5 – [INFORMATIVE] Interpretation as overlapping tables.....	26
Figure 6 – [INFORMATIVE] APO and real objects, non obvious possible interpretation.....	26
Figure 7 – ASE service conveyance.....	28
Figure 8 – Client/server confirmed interaction.....	29
Figure 9 – Client/server AR confirmed service primitives (positive case).....	30
Figure 10 – Client/server AR confirmed service primitives (negative case).....	30
Figure 11 – Client/server unconfirmed interaction.....	31

Figure 12 – Client/server AR unconfirmed service primitives	31
Figure 13 – Publish/subscribe communications stacks	32
Figure 14 – Publish/subscribe data-centric exchanges between decoupled network objects	33
Figure 15 – Publish/subscribe APOs services conveyed by the FAL	34
Figure 16 – [INFORMATIVE] Examples of publish/subscribe configurable behaviors via QoS	35
Figure 17 – Pull model interactions	37
Figure 18 – Push model interactions	38
Figure 19 – Publish/subscribe model interactions	39
Figure 20 – FAL ASEs	55
Figure 21 – Client/server encapsulated interface mechanism	102
Figure 22 – Publish/subscribe class derivations and relationships	117
Figure 23 – FAL ASEs and classes	118
Figure 24 – Publish/subscribe service request composition	128
Table 1 – Common client/server APOs	25
Table 2 – Class identification	48
Table 3 – Assigned vendor IDs	49
Table 4 – Filter service parameters	57
Table 5 – Read discretes service parameters	59
Table 6 – Read coils service parameters	63
Table 7 – Write single coil service parameters	64
Table 8 – Write multiple coils service parameters	66
Table 9 – Broadcast write single coil service parameters	67
Table 10 – Broadcast write multiple coils service parameters	68
Table 11 – Read input registers service parameters	71
Table 12 – Read holding registers service parameters	76
Table 13 – Write single holding register service parameters	78
Table 14 – Write multiple holding registers service parameters	79
Table 15 – Mask write holding register service parameters	81
Table 16 – Read/write holding registers service parameters	83
Table 17 – Read FIFO service parameters	85
Table 18 – Broadcast write single holding register service parameters	86
Table 19 – Broadcast write multiple holding registers service parameters	87
Table 20 – Read file service parameters	94
Table 21 – Write file service parameters	98
Table 22 – Device identification categories	104
Table 23 – Read device ID code	105
Table 24 – Conformity level	106
Table 25 – Requested vs. returned known objects	107
Table 26 – Read device identification service parameters	109
Table 27 – FAL class summary	115
Table 28 – Services by AREP role	116

Table 29 – Issue service parameters	120
Table 30 – Heartbeat service parameters.....	121
Table 31 – VAR service parameters	123
Table 32 – VAR service parameters	125
Table 33 – ACK service parameters	127
Table 34 – Header service parameters	130
Table 35 – INFO_DST service parameters	131
Table 36 – INFO_REPLY service parameters.....	132
Table 37 – INFO_SRC service parameters.....	134
Table 38 – INFO_TS service parameters	135
Table 39 – PAD service parameters	136
Table 40 – FAL class summary	138
Table 41 – Services by AREP role and sub-role	138

Withdrawn

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**INDUSTRIAL COMMUNICATION NETWORKS –
FIELDBUS SPECIFICATIONS –****Part 5-15: Application layer service definition – Type 15 elements**

FOREWORD

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

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

International Standard IEC 61158-5-15 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This bilingual version (2014-06) corresponds to the English version, published in 2007-12.

This first edition and its companion parts of the IEC 61158-5 subseries cancel and replace IEC 61158-5:2003. This edition of this part constitutes a technical revision. This part and its Type 15 companion parts also cancel and replace IEC/PAS 62030, published in 2004.

This edition of IEC 61158-5 includes the following significant changes from the previous edition:

- a) deletion of the former Type 6 fieldbus for lack of market relevance;
- b) addition of new types of fieldbuses;
- c) partition of part 5 of the third edition into multiple parts numbered -5-2, -5-3, ...

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

FDIS	Report on voting
65C/475/FDIS	65C/486/RVD

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

The French version of this standard has not been voted upon.

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

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

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

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

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

Withdrawn

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.

Withdrawn

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 behaviour 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 define 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

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

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

This 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) 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 (Ed.2.0), *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

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

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

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

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
- 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 behaviour 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 outputs

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

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

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, 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.

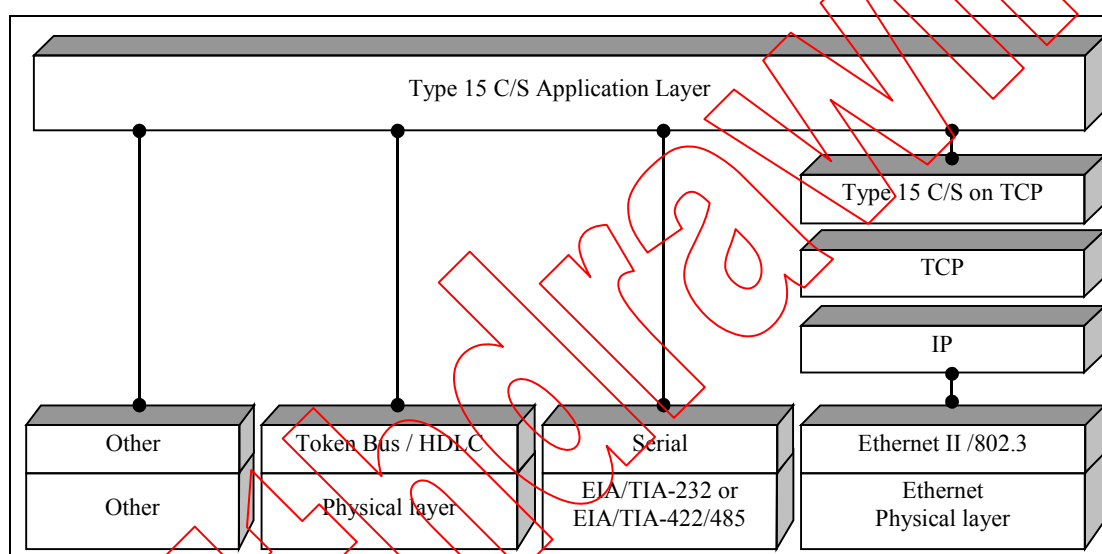


Figure 1 – Client/server stacks

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

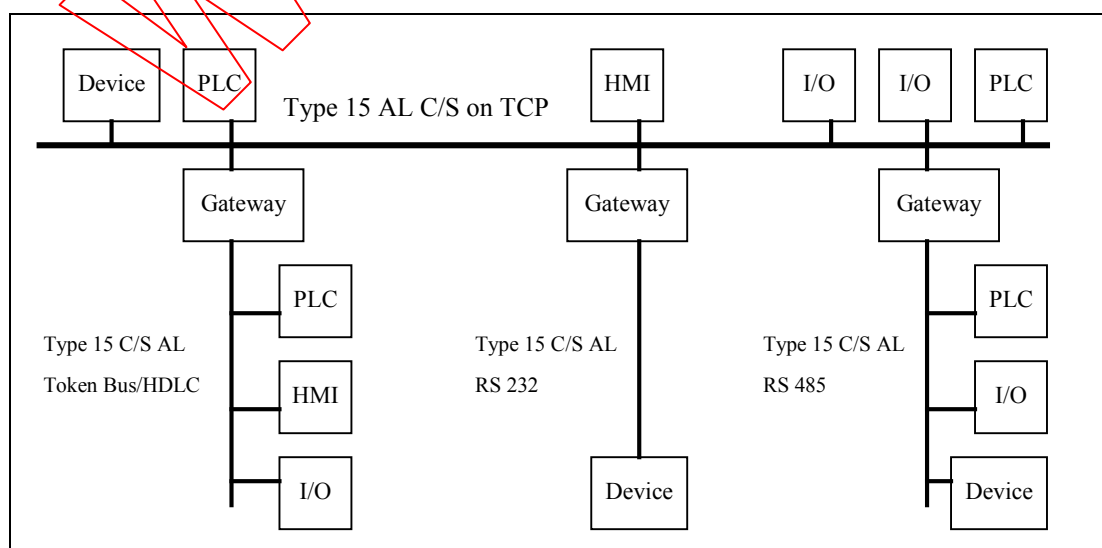


Figure 2 – Client/server communication on different buses or networks

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).

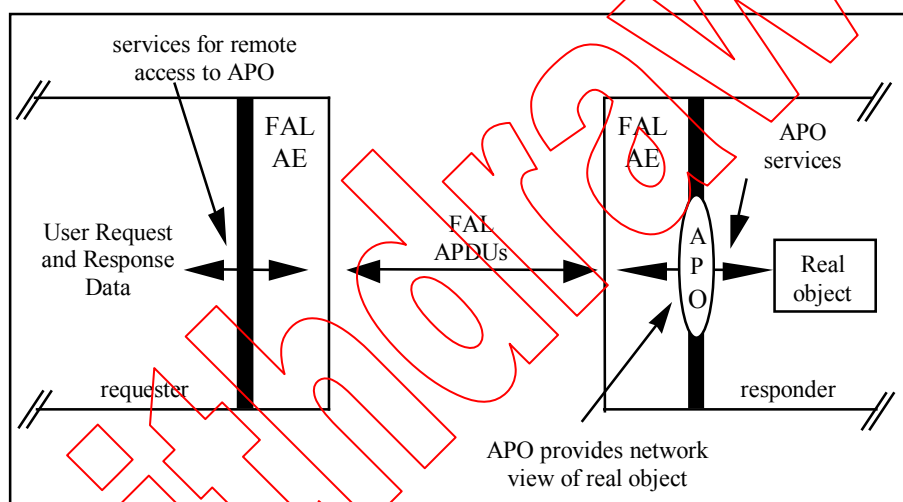


Figure 3 – Client/server APOs services conveyed by the FAL

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

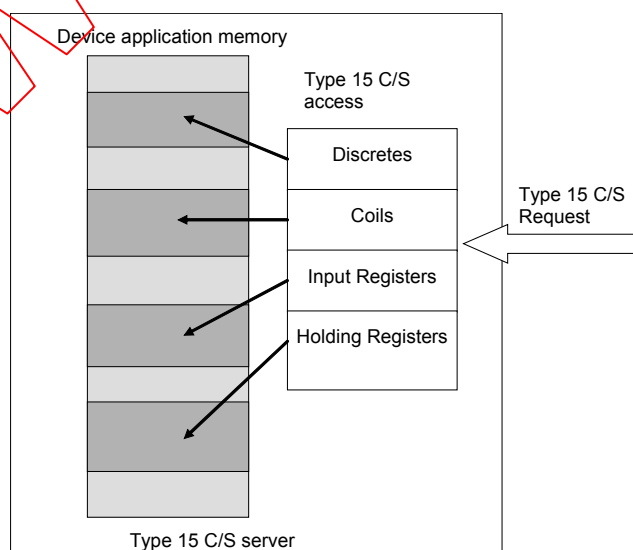
Client/server bases its model on APOs that have distinguishing characteristics. Some of them, namely discretes, coils, input registers and holding registers, are often interpreted as memory tables. While this is a simple and convenient model, it is just one of many, indeed they do not even need to be 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 – [INFORMATIVE] Interpretation as distinct tables**

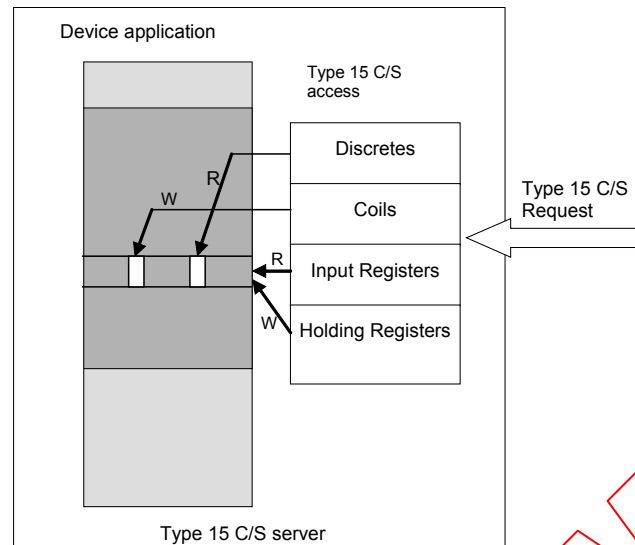


Figure 5 – [INFORMATIVE] Interpretation as overlapping tables

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.

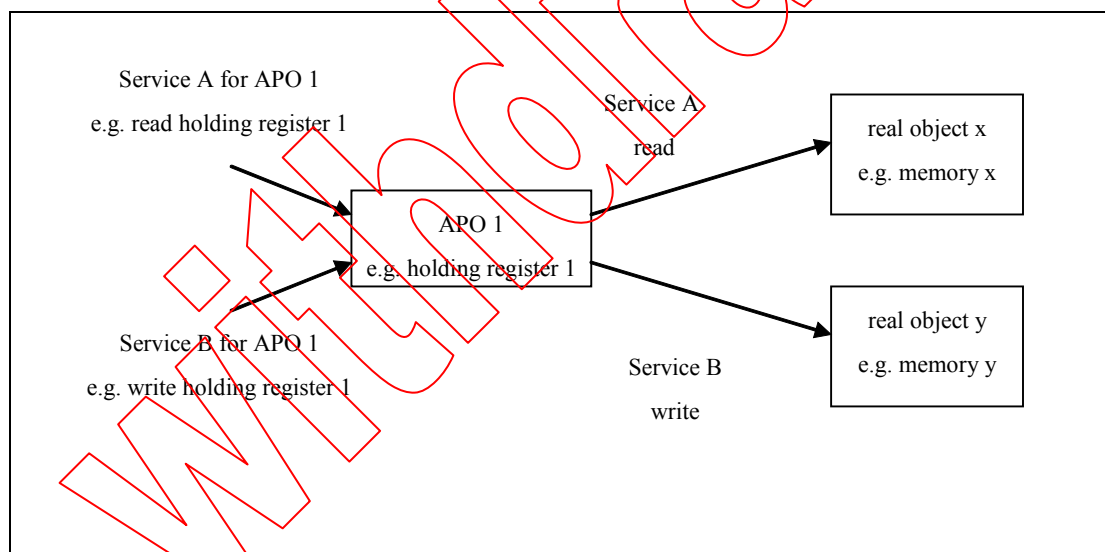


Figure 6 – [INFORMATIVE] APO and real objects, non obvious possible interpretation

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 65536 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 must 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.

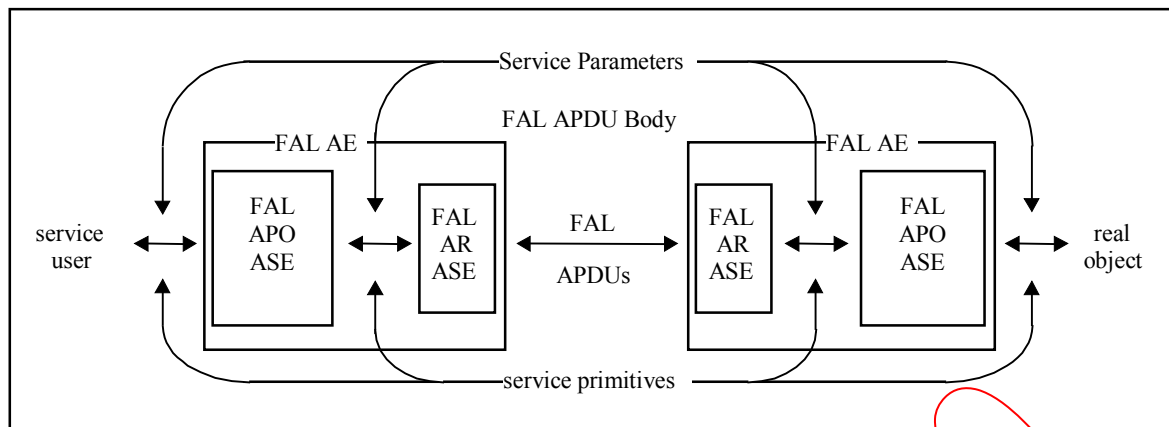


Figure 7 – ASE service conveyance

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 must 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.

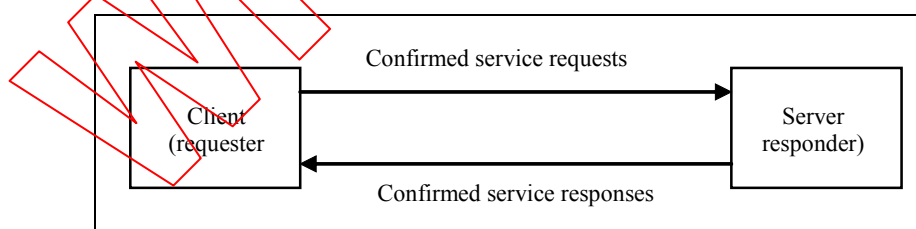


Figure 8 – Client/server confirmed interaction

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.

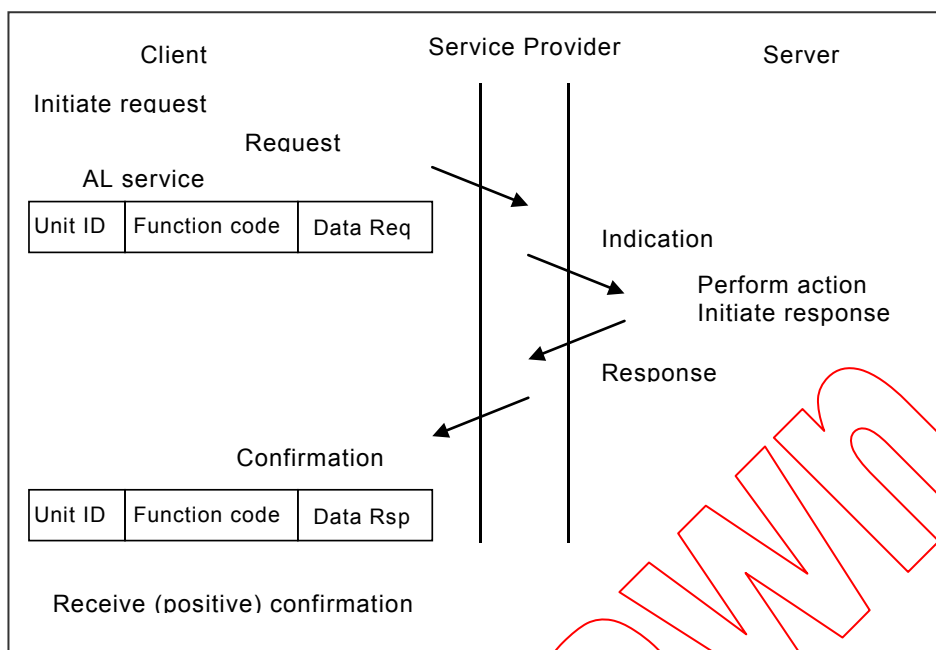


Figure 9 – Client/server AR confirmed service primitives (positive case)

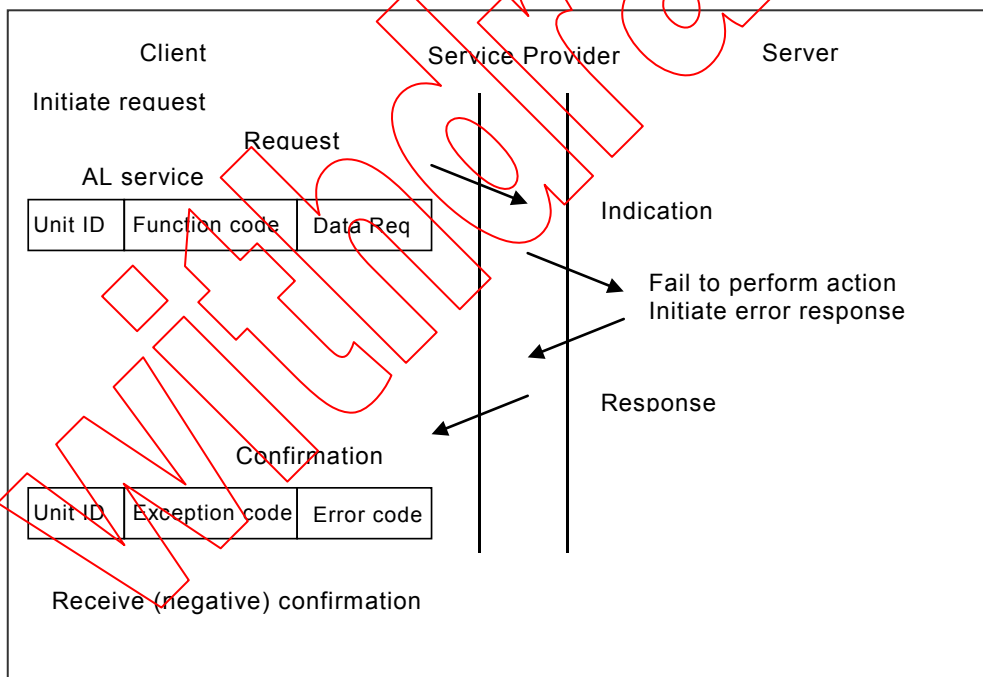


Figure 10 – Client/server AR confirmed service primitives (negative case)

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.

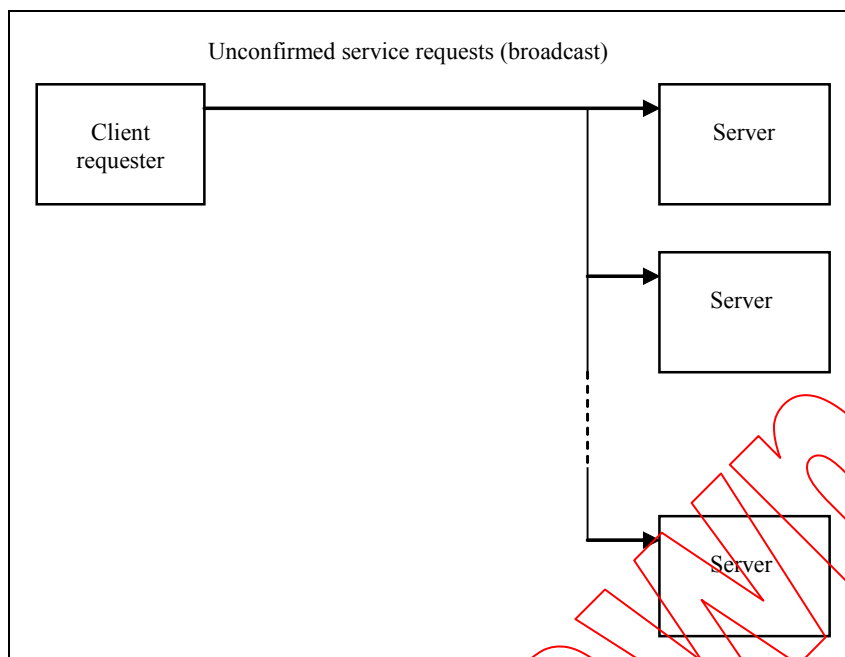


Figure 11 – Client/server unconfirmed interaction

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

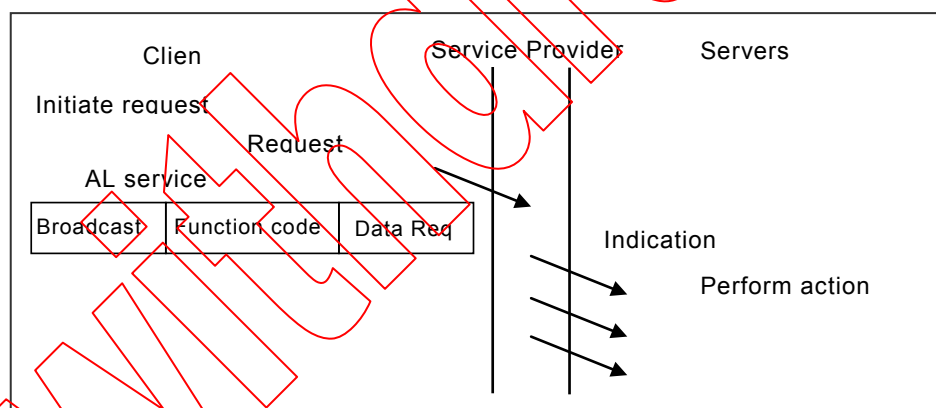


Figure 12 – Client/server AR unconfirmed service primitives

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.

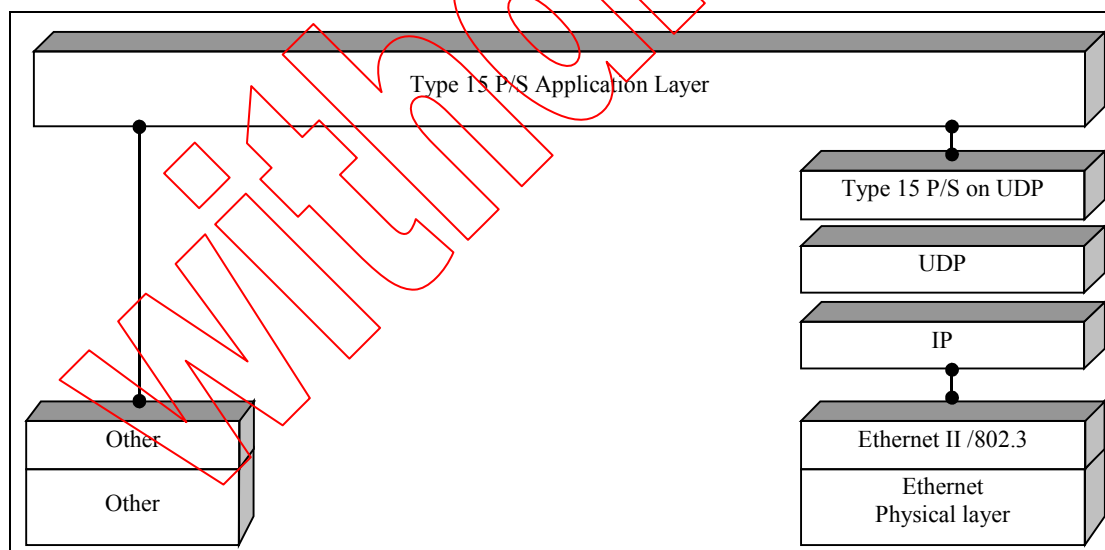


Figure 13 – Publish/subscribe communications stacks

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.

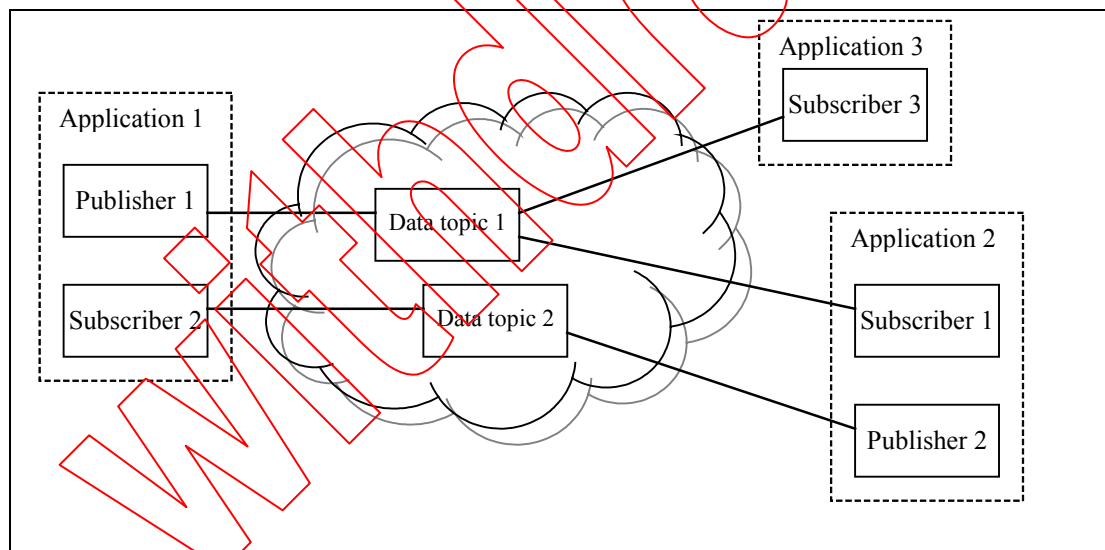


Figure 14 – Publish/subscribe data-centric exchanges between decoupled network objects

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 other

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.

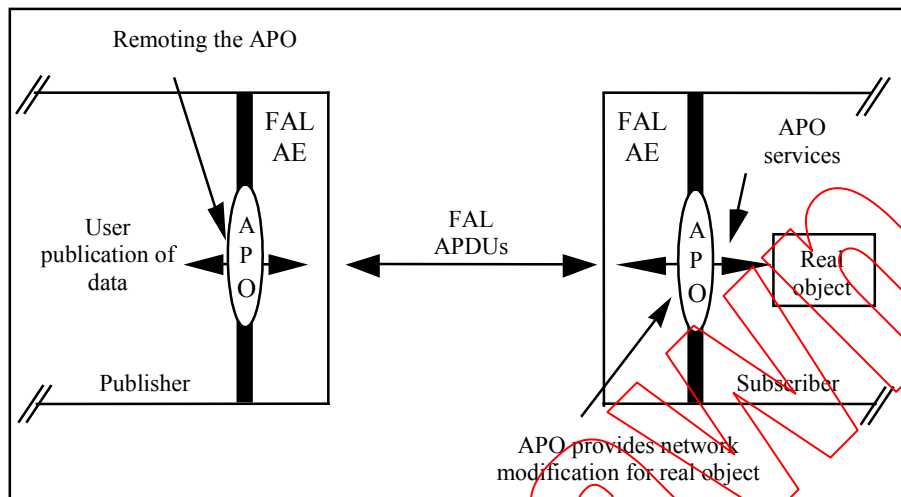


Figure 15 – Publish/subscribe APOs services conveyed by the FAL

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 must 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 must 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 must 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.

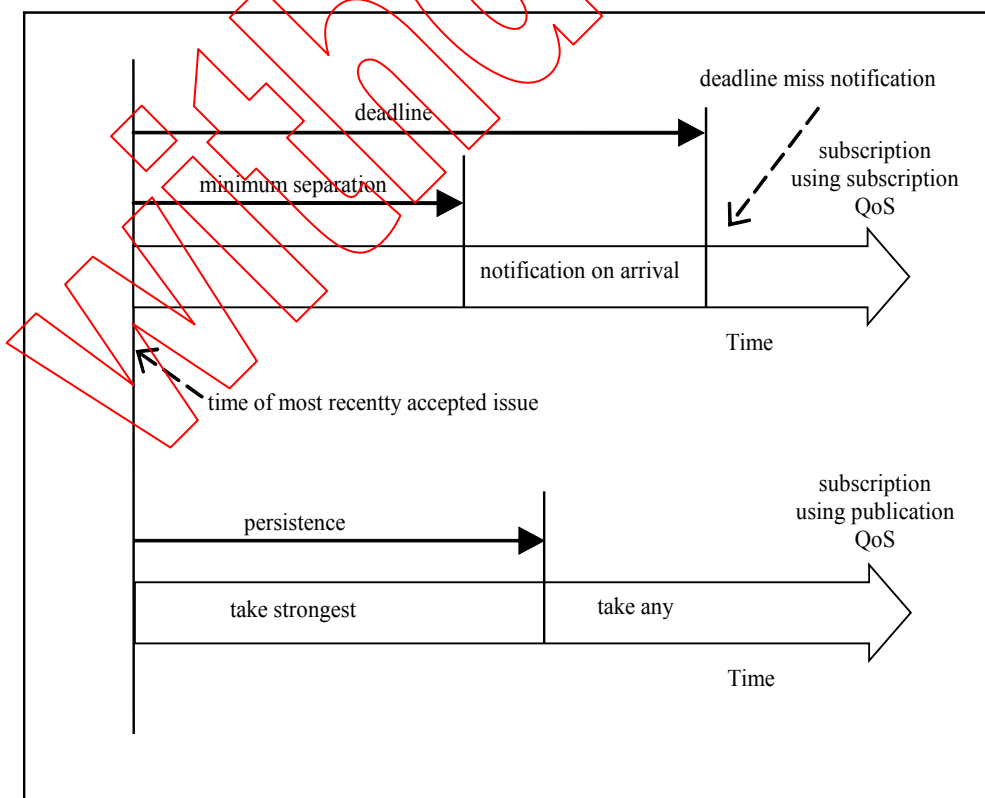


Figure 16 – [INFORMATIVE] Examples of publish/subscribe configurable behaviors via QoS

4.3.4 Cardinality and fault tolerance

Publish/subscribe exchanges can be

- one to one;
- one to many;
- many to one;
- many to many.

The “many to ..” possibilities directly support high availability with transparent failover.

4.3.5 Queues and reliability

Queues and acknowledgements configurable parameters allow an application to behave from best effort to strictly reliable.

4.3.6 Publish/subscribe communication model overview

4.3.6.1 General

Publish/subscribe interactions are based on data/events sent by one AP for use by others. This model is referred to as publisher/subscriber interactions.

The services supported by an interaction model are conveyed by application relationship endpoints (AREPs) associated with the communicating APs. The role that the AREP plays in the interaction (e.g. client, server, peer, publisher, subscriber) is defined as an attribute of the AREP.

While the general communication model architected around the APO exchanges is indeed that of publisher/subscriber interactions exchanging data topics, publish/subscribe AL supports the model using several unconfirmed services. Some publish/subscribe unconfirmed interactions do demand confirmation, but the confirmation is deferred to the AL user. From a communications perspective, there is no relationship between separate invocations of unconfirmed services as there is between the request and response of a confirmed service.

NOTE OMG DDS as in “Data Distribution Service for Real-Time Systems Specification, Version 1.1, December 2005” defines services that are here presented as responsibility of the publish/subscribe AL user instead. publish/subscribe is a wire protocol, and as such some functionality is deferred to the user application. This can be appreciated by the flexibility offered for implementations where foot-print is at a premium, and contributes to the pervasiveness of the approach.

4.3.6.2 Publisher/subscriber interactions

Common publisher/subscriber interactions involve a single publisher AP, and a group of one or more subscriber APs. This type of interaction has been defined to support variations of two models of interaction between APs, the “pull” model and the “push” model. In both models, the setup of the publishing AP is performed by management and is outside the scope of this standard.

4.3.6.2.1 Pull model interactions

In the “pull” model, the publisher receives a request to publish from a remote publishing *manager*, and broadcasts (or multicasts) its response across the network. The publishing manager is responsible only for initiating publishing by sending a request to the publisher.

Subscribers wishing to receive the published data listen for responses transmitted by the publisher. In this fashion, data is “pulled” from the publisher by requests from the publishing manager.

Confirmed FAL services are used to support this type of interaction. Two characteristics of this type of interaction differentiate it from the other types of interaction. First, a typical confirmed request/response exchange is performed between publishing manager and the publisher. However, the underlying conveyance mechanism provided by the FAL returns the response not just to the publishing manager, but also to all subscribers wishing to receive the published information. This is accomplished by having the Data Link Layer transmit the response to a group address, rather than to the individual address of the publishing manager. Therefore, the response sent by the publisher contains the published data and is multicast to the publishing manager and to all subscribers.

The second difference occurs in the behavior of the subscribers. Pull model subscribers, referred to as pull subscribers, are capable of accepting published data in confirmed service responses without having issued the corresponding request. Figure 17 illustrates these concepts.

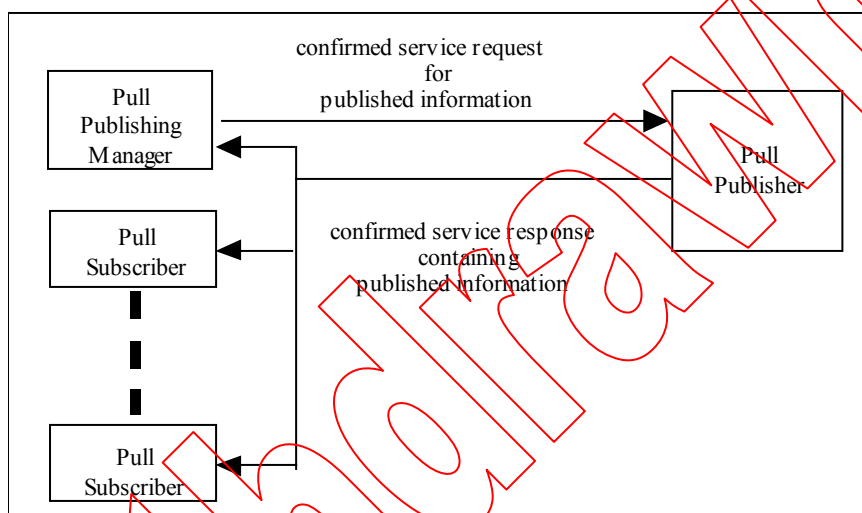


Figure 17 – Pull model interactions

4.3.6.2.2 Push model interactions

In the "push" model, two services may be used, one confirmed and one unconfirmed. The confirmed service is used by the subscriber to request to join the publishing. The response to this request is returned to the subscriber, following the client/server model of interaction. This exchange is only necessary when the subscriber and the publisher are located in different APs.

The unconfirmed service used in the Push Model is used by the publisher to distribute its information to subscribers. In this case, the publisher is responsible for invoking the correct unconfirmed service at the appropriate time and for supplying the appropriate information. In this fashion, it is configured to "push" its data onto the network.

Subscribers for the Push Model receive the published unconfirmed services distributed by publishers. Figure 18 illustrates the concept of the Push Model.

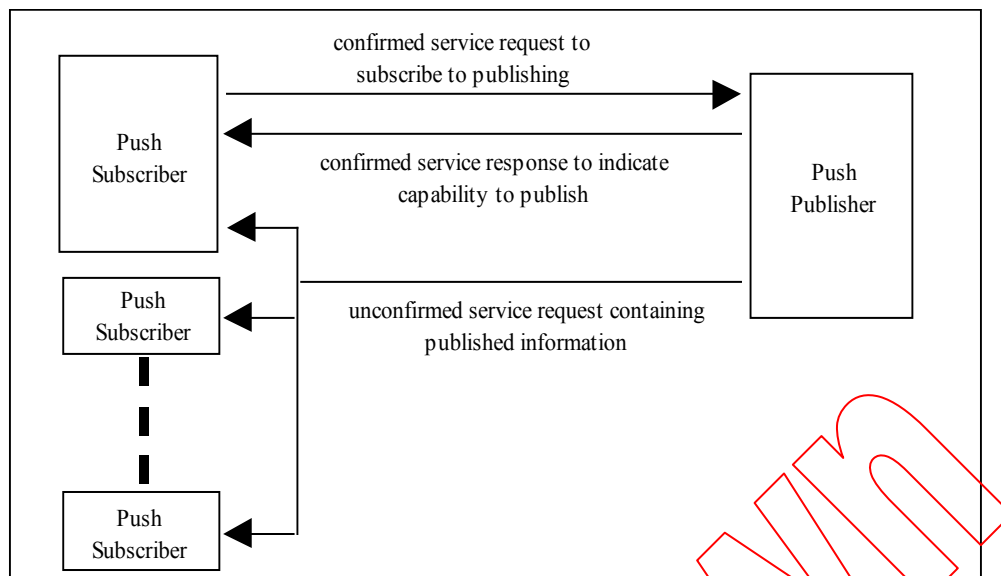


Figure 18 – Push model interactions

4.3.6.3 Publish/subscribe interactions

Publish/subscribe interactions are a variation of the push model interactions, this is illustrated via callouts in Figure 19.

Withdrawn

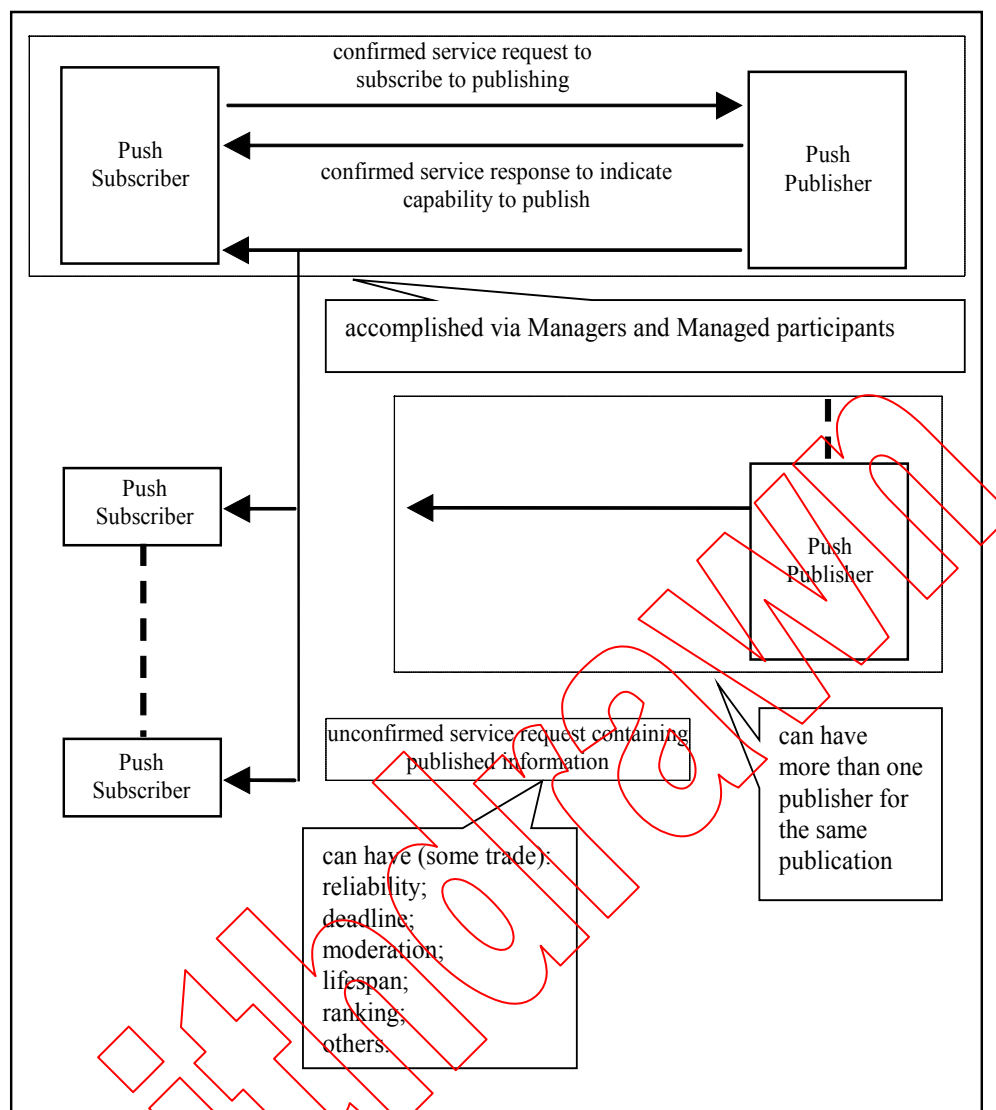


Figure 19 – Publish/subscribe model interactions

Managers and managed participants are publish/subscribe specialized applications, both containing publishers and subscribers or specialized versions of them.

The distinction between managers and managed participants is due to their role in the publish/subscribe discovery and maintenance mechanism. This mechanism allows publishing offers and subscription demands to topics to transparently match and properly behave even when publishers and subscribers come and go, dynamically.

Publish/subscribe publishers and subscribers, and their specialized versions, are collectively called communication actors. Their presence and attributes are the end result of the discovery and maintenance mechanism discussed.

It should be realized that the importance of the discovery and maintenance mechanism is in its accomplishments. Its optimality depends on the AL underlying layers, number of nodes and other hosting system features, therefore the discovery and maintenance mechanism described in this specification is just one of many, and as long as the end result is preserved, it can be replaced.

5 Data type ASE

5.1 General

All of IEC/TR 61158-1, 10.1 is incorporated by reference.

5.2 Formal definition of data type objects

All of IEC/TR 61158-1, 10.2 is incorporated by reference.

5.3 FAL defined data types

5.3.1 Common

5.3.1.1 Fixed length types

5.3.1.1.1 Boolean types

5.3.1.1.1.1 Boolean

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

This data type expresses a Boolean data type with the values TRUE and FALSE.

5.3.1.1.2 Bitstring types

Common not used.

5.3.1.1.3 Currency types

Common not used.

5.3.1.1.4 Date/Time types

Common not used.

5.3.1.1.5 Enumerated types

Common not used.

5.3.1.1.6 Handle types

Common not used.

5.3.1.1.7 Numeric types

Common not used.

5.3.1.1.7.1 Floating Point types

Common not used.

5.3.1.1.7.2 Integer types

5.3.1.1.7.2.1 long

This data type is the same as Integer32.

5.3.1.1.7.2.2 Integer32**CLASS:** Data Type**ATTRIBUTES:**

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

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

5.3.1.1.7.3 Unsigned types**5.3.1.1.7.3.1 Unsigned8****CLASS:** Data Type**ATTRIBUTES:**

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

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

5.3.1.1.7.3.2 USINT

This IEC 61131-3 type is the same as Unsigned8.

5.3.1.1.7.3.3 unsigned char

This data type is the same as Unsigned8.

5.3.1.1.7.3.4 Unsigned16**CLASS:** Data Type**ATTRIBUTES:**

1	Data Type Numeric Identifier	=	6
2	Data Type Name	=	Unsigned16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This type is a binary number. The most significant bit of the most significant byte is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of two octets.

5.3.1.1.7.3.5 UINT

This IEC 61131-3 type is the same as Unsigned16.

5.3.1.1.7.3.6 Unsigned32**CLASS:** Data Type**ATTRIBUTES:**

1	Data Type Numeric Identifier	=	7
2	Data Type Name	=	Unsigned32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type is a binary number. The most significant bit of the most significant byte is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of four octets.

5.3.1.1.7.3.7 WORD

This type is used in the same way as UINT.

5.3.1.1.8 Pointer types

Common not used.

5.3.1.1.9 OctetString types

5.3.1.1.9.1 OctetString

CLASS:	Data Type
ATTRIBUTES:	
1 Data Type Numeric Identifier	= 10
2 Data Type Name	= OctetString
3 Format	= STRING
4.1 Octet Length	= 1 to n

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet. IEC 61158-6 defines the order of transmission.

5.3.1.1.10 VisibleString character types

Common not used.

5.3.1.2 String types

Common not used.

5.3.1.3 Structure types

Common not used.

5.3.2 FAL defined data types for client/server

5.3.2.1 Fixed length types

5.3.2.1.1 Boolean types

No type specific boolean types.

5.3.2.1.2 Bitstring types

5.3.2.1.2.1 Status flag

CLASS:	Data Type
ATTRIBUTES:	
1 Data Type Numeric Identifier	= Not used
2 Data Type Name	= Status flag
3 Format	= FIXED LENGTH
4.1 Octet Length	= 2

This data type expresses a Status data type, with the values ON and OFF, encoded in two octets. The encoding for ON is 0xFF00 and the encoding for OFF is 0x0000.

5.3.2.1.2.2 Status bit sequence**CLASS:** Data Type**ATTRIBUTES:**

- | | | | |
|-----|------------------------------|---|---------------------|
| 1 | Data Type Numeric Identifier | = | Not used |
| 2 | Data Type Name | = | Status bit sequence |
| 3 | Format | = | STRING |
| 4.1 | Octet Length | = | 1 to n |

This data type expresses a Status bit sequence data type, with the values ON and OFF, encoded one value per bit. The encoding for ON is 1 and the encoding for OFF is 0. The bits are contained in octets, with the last octet 0-padded if the number of bits in the sequence is not a multiple of 8. The bits are numbered in the order as reported below, where y is a Status bit number, x is a Status bit value and – is an empty location:

	MSB							LSB
Bit	8	7	6	5	4	3	2	1
Octet 1	7	6	5	4	3	2	1	0
Status	x	x	x	x	x	x	x	x
Octet 2	15	14	13	12	11	10	9	8
Status	x	x	x	x	x	x	x	x
...								
...								
Octet n	-	-	-	-	-	y	y	y
Status	0	0	0	0	0	x	x	x

5.3.2.1.3 Currency types

No type specific currency types.

5.3.2.1.4 Date/Time types

No type specific date/time types.

5.3.2.1.5 Enumerated types

No type specific enumerated types.

5.3.2.1.6 Handle types

No type specific handle types.

5.3.2.1.7 Numeric types

No type specific general numeric types.

5.3.2.1.7.1 Floating Point types

No type specific floating point types.

5.3.2.1.7.2 Integer types

No type specific integer types.

5.3.2.1.7.3 Unsigned types

No type specific unsigned types.

5.3.2.1.8 Pointer types

No type specific pointer types.

5.3.2.1.9 OctetString types

No type specific fixed length octet (byte) string types.

5.3.2.1.10 VisibleString character types

No type specific fixed length visible string types.

5.3.2.2 String types

5.3.2.2.1 ASCII string

CLASS:

Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	Not used
2	Data Type Name	=	ASCII string
3	Format	=	STRING
4.1	Octet Length	=	1 to n

5.3.2.3 Structure types

No type specific structure types.

5.3.3 FAL defined data types for publish/subscribe

5.3.3.1 Fixed length types

5.3.3.1.1 Boolean types

No type specific boolean types.

5.3.3.1.2 Bitstring types

No type specific bitstring types.

5.3.3.1.3 Currency types

No type specific currency types.

5.3.3.1.4 Date/Time types

No type specific date/time types.

5.3.3.1.5 Enumerated types

No type specific enumerated types.

5.3.3.1.6 Handle types

No type specific handle types.

5.3.3.1.7 Numeric types

No type specific general numeric types.

5.3.3.1.7.1 Floating Point types

No type specific floating point types.

5.3.3.1.7.2 Integer types

No type specific integer types.

5.3.3.1.7.3 Unsigned types

5.3.3.1.7.3.1 IPAddress

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	Not used
2	Data Type Name	=	IPAddress
3	Format	=	Unsigned32

An IP address is a 4-octets unsigned number, as from RFC 791.

An IP address of zero is an invalid IP address.

IPADDRESS_INVALID 0

The mapping between the dot-notation "a.b.c.d" of an IP address and its representation as an unsigned long is as follows:

$$\text{IPAddress ipAddress} = (((a * 256 + b) * 256) + c) * 256 + d$$

Example: IP address "127.0.0.1" corresponds to the unsigned long number 2130706433 or 0x7F000001.

5.3.3.1.7.3.2 Port

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	Not used
2	Data Type Name	=	Port
3	Format	=	Unsigned32

A port is a 4-octets unsigned number.

The port number zero is an invalid port-number

PORT_INVALID 0

If a port number represents an IPv4 UDP port, only the range of unsigned short numbers from 0x1 to 0x0000ffff is valid.

The following ports have been assigned to publish/subscribe by IANA:

— rtps-discovery	7400/tcp,udp	RTPS Discovery
— rtps-dd-ut	7401/tcp,udp	RTPS Data Distribution User Traffic
— rtps-dd-mt	7402/tcp,udp	RTPS Data Distribution Meta Traffic

5.3.3.1.8 Pointer types

No type specific pointer types.

5.3.3.1.9 OctetString types

No type specific fixed length octet (byte) string types.

5.3.3.1.10 VisibleString character types

No type specific fixed length visible string types.

5.3.3.2 String types

No type specific string types.

5.3.3.3 Structure types

5.3.3.3.1 HostID

CLASS:		Data Type
ATTRIBUTES:		
1	Data Type Numeric Identifier	= Not used
2	Data Type Name	= HostID
3	Format	= STRUCTURE
4	Number of Fields	= 1
5.1	Field Name	= hostID
5.1.1	Field Data Type	= OctetString
5.1.2	Field Length	= 4

The HostID is actually not structured at all, but just an OctetString. The distinction can be ignored for all practical purposes.

The value { 0x00, 0x00, 0x00, 0x00 } is reserved for HOSTID_UNKNOWN with the meaning of unknown HostID.

5.3.3.3.2 AppID

CLASS:		Data Type
ATTRIBUTES:		
1	Data Type Numeric Identifier	= Not used
2	Data Type Name	= AppID
3	Format	= STRUCTURE
4	Number of Fields	= 2
5.1	Field Name	= instanceID
5.1.1	Field Data Type	= OctetString
5.1.2	Field Length	= 3
5.2	Field Name	= appKind
5.2.1	Field Data Type	= OctetString
5.2.2	Field Length	= 1
5.2.3	Field Content	= 0x01 or 0x02

An appKind value of 0x01 tags the application as a publish/subscribe managed participant.

An appKind value of 0x02 tags the application as a publish/subscribe manager.

described in this specification is just one of many, and as long as the end result is preserved, it can be replaced.

Every manager and every managed participant (or managed application) contains a number of special built-in network objects, which have reserved Object IDs.

These special objects fall into these categories:

- the domain participant (application) itself is a network object with a special GUID (the instance of the application is called *applicationSelf*). In addition, every application has a CSTWriter (*writerApplicationSelf*) that disseminates the attributes of the local application on the network;
- Several objects are dedicated to the discovery of managers and managed participants (managed applications) on the network. Every managed application has the CSTReaders *readerApplications* and *readerManagers*, through which the existence and attributes of the remote managed applications and remote managers, respectively, are obtained. Every manager has the corresponding CSTWriters *writeApplications* and *writeManagers*;
- Every managed application has, among others, two instances of a CSTReader (*readerPublications* and *readerSubscriptions*) and two instances of a CSTWriter (*writerPublications* and *writerSubscriptions*). Through the CSTReaders, the managed application can receive information about the existence and attributes of all the remote publications and subscriptions in the network. Through the CSTWriters, the managed application can send out information about its local publications and subscriptions.

5.3.3.3.2 Classes

The last six bits of the objectId define the class to which the object belongs (application, Publisher, Subscriber, CSTWriter, or CSTReader). Table 2 provides an overview. The meaning of the message IDs is fixed in this major version (1). New *objKinds* may be added in higher minor versions as the publish/subscribe object-model is extended with new classes.

Table 2 – Class identification

Class of object	Normal user-object	Reserved user-object	Normal meta-object	Reserved meta-object
unknown	0x00	0x40	0x80	0xc0
application	0x01	0x41	0x81	0xc1
CSTWriter	0x02	0x42	0x82	0xc2
Publisher	0x03	0x43	0x83	0xc3
Subscriber	0x04	0x44	0x84	0xc4
CSTReader	0x07	0x47	0x87	0xc7

5.3.3.3.4 GUID

CLASS:

Data Type

ATTRIBUTES:

- | | | | |
|-------|------------------------------|---|-----------|
| 1 | Data Type Numeric Identifier | = | Not used |
| 2 | Data Type Name | = | GUID |
| 3 | Format | = | STRUCTURE |
| 4 | Number of Fields | = | 3 |
| 5.1 | Field Name | = | hostID |
| 5.1.1 | Field Data Type | = | HostID |
| 5.1.2 | Field Length | = | 4 |
| 5.2 | Field Name | = | applID |
| 5.2.1 | Field Data Type | = | AppID |
| 5.2.2 | Field Length | = | 4 |
| 5.3 | Field Name | = | objectId |