

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Industrial communication networks – Fieldbus specifications –
Part 5-21: Application layer service definition – Type 21 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 5-21: Définition des services de la couche application – Éléments
de Type 21**

Withdrawing



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2010 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 la CEI ou du Comité national de la CEI du pays du demandeur.

Si vous avez des questions sur le copyright de la CEI ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de la CEI de votre pays de résidence.

IEC Central Office
3, rue de Varembé
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.

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you 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It also gives information on projects, replaced and withdrawn publications.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 on-line 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 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

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 la CEI

La Commission Electrotechnique Internationale (CEI) 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 CEI

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

Liens utiles:

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

La recherche avancée vous permet de trouver des publications CEI 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.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. 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 au monde 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 les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

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 5-21: Application layer service definition – Type 21 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 5-21: Définition des services de la couche application – Éléments
de Type 21**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XB

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-88912-865-5

**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.....	4
INTRODUCTION.....	6
1 Scope.....	7
1.1 Overview.....	7
1.2 Specifications.....	8
1.3 Conformance.....	8
2 Normative references.....	8
3 Terms, definitions, symbols, abbreviations, and conventions.....	9
3.1 Terms and definitions from other ISO/IEC standards.....	9
3.2 Fieldbus data link layer terms.....	9
3.3 Fieldbus application layer specific definitions.....	10
3.4 Abbreviations and symbols.....	16
3.5 Conventions.....	16
4 Concepts.....	19
4.1 Common concepts.....	19
4.2 Type specific concepts.....	36
5 Data type ASE.....	39
5.1 General.....	39
5.2 Formal definition of data type objects.....	42
5.3 FAL defined data types.....	43
5.4 Data type ASE service specification.....	47
6 Communication model specification.....	47
6.1 ASEs.....	47
6.2 ARs.....	68
6.3 Summary of FAL classes.....	71
6.4 Permitted FAL services by AREP role.....	71
Bibliography.....	73
Figure 1 – Relationship to the OSI Basic Reference Model.....	20
Figure 2 – Architectural positioning of the fieldbus application layer.....	20
Figure 3 – Client/server interactions.....	23
Figure 4 – Pull model interactions.....	24
Figure 5 – Push model interactions.....	24
Figure 6 – APOs services conveyed by the FAL.....	26
Figure 7 – Application entity structure.....	28
Figure 8 – FAL management of objects.....	29
Figure 9 – ASE service conveyance.....	30
Figure 10 – Defined and established AREPs.....	32
Figure 11 – FAL architectural components.....	34
Figure 12 – Interaction between FAL and DLL.....	37
Figure 13 – Publisher-subscriber communication model.....	37
Figure 14 – Client-server communication model.....	38
Figure 15 – Object model.....	38
Figure 16 – ASEs of a Type 21 application.....	38

Figure 17 – Data type class hierarchy example	40
Figure 18 – The AR ASE conveys APDUs between APs.....	61
Table 1 – Types of timeliness	25
Table 2 – Overall structure of the OD	38
Table 3 – Identify service	50
Table 4 – Status service	52
Table 5 – Access rights for object	54
Table 6 – Read service	55
Table 7 – Write service	57
Table 8 – TB-transfer	60
Table 9 – COS-transfer	60
Table 10 – Conveyance of service primitives by AREP role.....	62
Table 11 – Valid combinations of AREP roles involved in an AR	62
Table 12 – AR-unconfirmed send	66
Table 13 – AR-confirmed send.....	67
Table 14 – FAL class summary	71
Table 15 – Services by AREP role	72

Withdawn

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**INDUSTRIAL COMMUNICATION NETWORKS –
FIELDBUS SPECIFICATIONS –****Part 5-21: Application layer service definition –
Type 21 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 itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 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 profile parts. Use of the various protocol types in other combinations may require permission of their respective intellectual-property-right holders.

International Standard IEC 61158-5-21:2010 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This standard cancels and replaces IEC/PAS 62573 published in 2008. This first edition constitutes a technical revision.

This bilingual version published in 2012-01 corresponds to the English version published in 2010-08.

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

FDIS	Report on voting
65C/606/FDIS	65C/620/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 has not been voted upon.

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

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

The committee has decided that the contents of this publication will remain unchanged until the stability 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.

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 5-21: Application layer service definition – Type 21 elements

1 Scope

1.1 Overview

The Fieldbus Application Layer (FAL) provides user programs with a means to access the fieldbus communication environment. In this respect, the FAL can be considered a window between corresponding application programs.

This standard provides the common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment as well as material specific to the Type 21 protocol. The term “time-critical” is used to represent the presence of a time-window within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant, and possibly human life.

This standard defines, in an abstract way, the externally visible service provided by the FAL in terms of:

- a) an abstract model for defining application resources (objects) capable of being manipulated by users via 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 that they take;
- d) the interrelationship between these actions and events, and their valid sequences.

The purpose of this standard is to define the services provided to:

- a) the FAL-user at the boundary between the user and the application layer of the fieldbus Reference Model;
- b) systems management at the boundary between the application layer and systems management of the fieldbus Reference Model.

This standard describes the structure and services of the IEC FAL, 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 (AEs) contained in 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 management of the instances of FAL classes.

Although these services specify how requests and responses are issued and delivered from the perspective of applications, they do not include a specification of what the requesting and responding applications are to do with them. That is, these services only define what requests and responses applications can send or receive, not the functions of the applications themselves. 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 standard 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. This latter objective 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 standard may be used as the basis for formal application programming interfaces. Nevertheless, it is not a formal programming interface, and any such interface must address implementation issues not covered by this standard, including:

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

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities in industrial automation systems.

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through the implementation of conforming application layer protocols that fulfill any given type of application layer services as defined in this standard.

2 Normative references

The following 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 60559, *Binary floating-point arithmetic for microprocessor systems*

IEC 61158-2:2010¹, *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-3-21:2010¹, *Industrial communication networks – Fieldbus specifications – Part 3-21: Data-link layer service definition – Type 21 elements*

IEC 61158-4-21:2010¹, *Industrial communication networks – Fieldbus specifications – Part 4-21: Data-link layer protocol specification – Type 21 elements*

IEC 61158-6-21:2010¹, *Industrial communication networks – Fieldbus specifications – Part 6-21: Application layer protocol specification – Type 21 elements*

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

¹ To be published.

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

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

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

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

3 Terms, definitions, symbols, abbreviations, and conventions

3.1 Terms and definitions from other ISO/IEC standards

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

3.2 Fieldbus data link layer terms

For the purposes of this document, the following terms as defined in IEC 61158-3-21:2010 and IEC 61158-4-21:2010 apply.

- a) DL-Time
- b) DL-Scheduling-policy
- c) DLCEP

- d) DLC
- e) DL-connection-oriented mode
- f) DL PDU
- g) DLSDU
- h) DLSAP
- k) link
- l) ISO/IEC 8802-3:2000 MAC address
- m) DL-entity identifier

3.3 Fieldbus application layer specific definitions

3.3.1 application

function or data structure for which data are consumed or produced

3.3.2 application objects

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

3.3.3 application process

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

3.3.4 application process identifier

distinguishes multiple application processes used in a device

3.3.5 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 (see the definition for "application process object class"). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions, and to create and delete application objects and their corresponding definitions dynamically.

3.3.6 application process object class

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

3.3.7 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.

3.3.8 application relationship application service element

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

3.3.9**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.

3.3.10**attribute**

description of an externally visible characteristic or feature of an object

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

3.3.11**behavior**

indication of how an object responds to particular events

3.3.12**channel**

single physical or logical link of an input or output application object of a server to the process

3.3.13**class**

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

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

3.3.14**class attributes**

attribute shared by all objects within the same class

3.3.15**class code**

unique identifier assigned to each object class

3.3.16**class-specific service**

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

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

3.3.17**client**

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

3.3.18**consume**

act of receiving data from a producer

3.3.19**consumer**

node or sink that receives data from a producer

3.3.20

consuming application

application that consumes data

3.3.21

conveyance path

unidirectional flow of APDUs across an application relationship

3.3.22

cyclic

repetitive in a regular manner

3.3.23

data consistency

means for coherent transmission and access of the input- or output-data object between and within client and server

3.3.24

device

physical hardware connected to the link

NOTE A device may contain more than one node.

3.3.25

device profile

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

3.3.26

diagnostic information

all data available at the server for maintenance purposes

3.3.27

end node

producing or consuming node

3.3.28

endpoint

one of the communicating entities involved in a connection

3.3.29

error

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

3.3.30

error class

general grouping for related error definitions and corresponding error codes

3.3.31

error code

identification of a specific type of error within an error class

3.3.32

event

instance of a change of conditions

3.3.33**FIFO variable**

variable object class composed of a set of homogeneously typed elements, where the first written element is the first element that can be read

NOTE In a fieldbus system, only one complete element can be transferred as a result of one service invocation.

3.3.34**frame**

simplified synonym for data link protocol data unit (DLPDU)

3.3.35**group**

- a) (General): a general term for a collection of objects
- b) (Addressing): when describing an address, an address that identifies more than one entity

3.3.36**invocation**

act of using a service or other resource of an application process

NOTE Each invocation represents a separate thread of control that may be described by its context. Once the service completes, or use of the resource is released, the invocation ceases to exist. For service invocations, a service that has been initiated but not yet completed is referred to as an outstanding service invocation. For service invocations, an Invoke ID may be used to identify the service invocation unambiguously and differentiate it from other outstanding service invocations.

3.3.37**index**

address of an object within an application process

3.3.38**instance**

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

EXAMPLE California is an instance of the object class US-state.

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

3.3.39**instance attributes**

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

3.3.40**instantiated**

object that has been created in a device

3.3.41**logical device**

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

3.3.42**manufacturer ID**

identification of each product manufacturer by a unique number

3.3.43

management information

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

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

3.3.44

network

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

3.3.45

object

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

3.3.46

object dictionary

collection of definitions, communication-specific attributes and parameters, and application-dependent data

3.3.47

object-specific service

service unique to the object class that defines it

3.3.48

physical device

automation or other network device

3.3.49

point-to-point connection

connection that exists between exactly two application objects

3.3.50

pre-established AR endpoint

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

3.3.51

process data

object(s) that are already pre-processed and transferred cyclically for the purpose of information or further processing

3.3.52

produce

act of sending data to be received by a consumer

3.3.53

producer

node that is responsible for sending data

3.3.54

property

general term for descriptive information about an object

3.3.55**provider**

source of a data connection

3.3.56**publisher**

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

NOTE A publisher may not be aware of the identity or number of subscribers.

3.3.57**publishing manager**

role of an AR endpoint in which it issues one or more confirmed service request application protocol data units (APDUs) to a publisher to request that a specified object be published. Two types of publishing managers are defined by this standard, pull publishing managers and push publishing managers, each of which is defined separately.

3.3.58**push publisher**

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

3.3.59**push publishing manager**

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

3.3.60**push subscriber**

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

3.3.61**server**

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

3.3.62**service**

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

3.3.63**station**

host of one AP, identified by a unique data link connection endpoint (DLCEP)-address

3.3.64**subscriber**

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

3.4 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
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 -
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
DNS	Domain Name Service
FAL	Fieldbus Application Layer
Ind	Indication
Req	Request
Rsp	Response

3.5 Conventions

3.5.1 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. Access to these attributes is beyond the scope of this document except where specified. The service specification defines the services provided by the ASE.

3.5.2 General conventions

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

3.5.3 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 as 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 not used for Type 21 elements.
- (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 indicates 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), or selector (s) indicator in column 2; an attribute type label in column 3; a name or a conditional expression in column 4; and an optional 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. The numbers below refer to the general template form above. Nesting is used to specify:
 - i) fields of a structured attribute (4.1, 4.2, 4.3);
 - ii) attributes conditional on a constraint statement. 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 a management service (2).
- d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

3.5.4 Conventions for service definitions

3.5.4.1 General

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

3.5.4.2 Service parameters

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

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

- a) parameter name;
- b) request primitive;
- c) indication primitive;
- d) response primitive;
- e) confirmation primitive.

One parameter, or a component, is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the column:

- M The parameter is mandatory for the primitive.
- U The 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 The parameter is conditional upon other parameters or upon the environment of the service user.
- (blank) The parameter is never present.
- S The parameter is a selected item.

Some entries are further qualified by items in parentheses. 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.

3.5.4.3 Service procedures

The service procedures are defined in terms of:

- a) The interactions between application entities through the exchange of fieldbus APDUs;
- b) 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 that support time-constrained communications services within the fieldbus application layer.

4 Concepts

4.1 Common concepts

4.1.1 Overview

The fieldbus is intended to be used in factories and process plants to interconnect primary automation devices (e.g., sensors, actuators, local display devices, annunciators, programmable logic controllers, small single loop controllers, and standalone field controls) with control and monitoring equipment located in control rooms.

Primary automation devices are associated with the lowest levels of the industrial automation hierarchy and perform a limited set of functions within a definite time window. Some of these functions include diagnostics, data validation, and handling of multiple inputs and outputs.

These primary automation devices, also called field devices, are located close to the process fluids, the fabricated part, the machine, the operator, and the environment. This use positions the fieldbus at the lowest levels of the computer integrated manufacturing (CIM) architecture.

Some of the expected benefits in using fieldbus systems are reductions in wiring, increases in the amount of data exchanged, a wider distribution of control between the primary automation devices and the control room equipment, and satisfaction of time-critical constraints.

This subclause describes the fundamentals of the FAL. Detailed descriptive information about each of the FAL ASEs can be found in the overview subclause of each of the communication model specifications.

4.1.2 Architectural relationships

4.1.2.1 Relationship to the application layer of the OSI Basic Reference Model

The functions of the FAL have been described according to OSI layering principles. However, the FAL's architectural relationship to the lower layers is different, as follows (see Figure 1):

- a) the FAL groups OSI functions together with extensions to cover time-critical requirements. The OSI application layer structure standard (ISO/IEC 9545) was used as a basis for the FAL specification;
- b) the FAL directly uses the services of the underlying layer. The underlying layer may be the DLL or any layer in between. When using the underlying layer, the FAL may provide functions normally associated with the OSI middle layers for proper mapping onto the underlying layer.

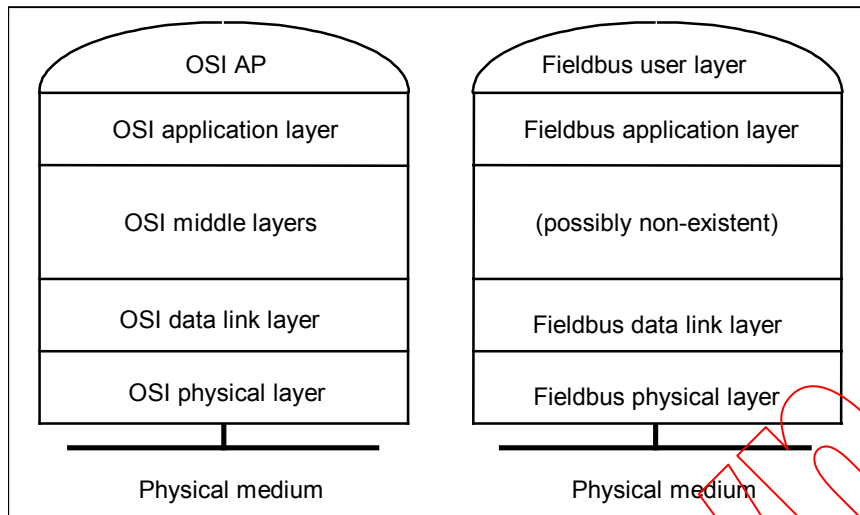


Figure 1 – Relationship to the OSI Basic Reference Model

4.1.2.2 Relationships to other fieldbus entities

4.1.2.2.1 General

The FAL architectural relationships illustrated in Figure 2 have been designed to support the interoperability needs of time-critical systems distributed in the fieldbus environment.

In this environment, the FAL provides communications services to time-critical and non-time-critical applications located in fieldbus devices.

In addition, the FAL uses the DLL directly to transfer its application layer protocol data units, using a set of data transfer services and a set of supporting services to control the operational aspects of the DLL.

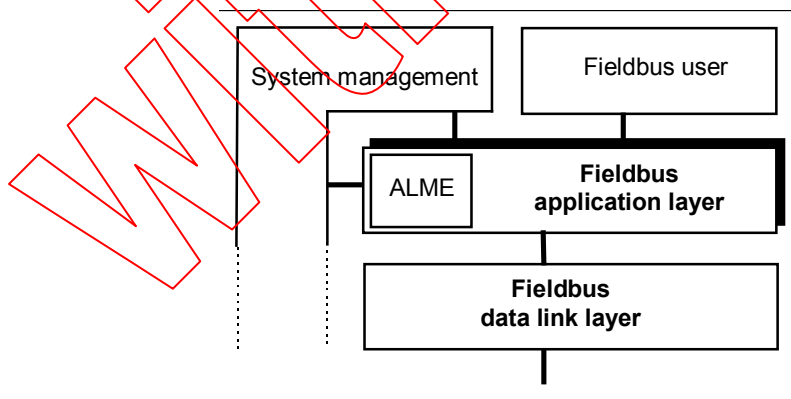


Figure 2 – Architectural positioning of the fieldbus application layer

4.1.2.2.2 Use of the fieldbus data link layer

The FAL provides network access to fieldbus APs. It interfaces directly to the fieldbus DLL for transfer of its APDUs.

The DLL provides various types of service to the FAL for transfer of data between data link endpoints (e.g., DLSAPs and DLCEPs).

4.1.2.2.3 Support for fieldbus applications

Fieldbus applications appear to the network as application processes (APs). APs are the components of a distributed system that may be individually identified and addressed.

Each AP contains an FAL AE that provides network access for the AP. That is, each AP communicates with other APs through its AE. In this sense, the AE provides a window of visibility into the AP.

APs contain identifiable components that are also visible across the network. These components are represented to the network as APOs. They may be identified by one or more key attributes. They are located at the address of the application process that contains them.

The services used to access the APOs are provided by APO-specific application service elements (ASEs) contained in the FAL. These ASEs are designed to support user, function block, and management applications.

4.1.2.2.4 Support for system management

The FAL services can be used to support various management operations, including management of fieldbus systems, applications, and the fieldbus network.

4.1.2.2.5 Access to FAL layer management entities

One LME may be present in each FAL entity on the network. fieldbus application layer management entities (FALMEs) provide access to the FAL for system management purposes.

The set of data accessible by the system manager is referred to as the system management information base (SMIB). Each FALME provides the FAL portion of the SMIB. Implementation of the SMIB is beyond the scope of this standard.

4.1.3 Fieldbus application layer structure

4.1.3.1 Overview

The structure of the FAL is a refinement of the OSI application layer structure (ISO/IEC 9545). As a result, the organization of this subclause is similar to that of ISO/IEC 9545. Certain concepts presented here have been refined from ISO/IEC 9545 for the fieldbus environment.

The FAL differs from the other layers of the OSI Basic Reference Model in the following two principal aspects:

- a) the OSI Basic Reference Model defines the association, a single type of application layer communications channel to connect APs to each other. The FAL defines the application relationship (AR), of which there are several types, to permit application processes (APs) to communicate with each other;
- b) the FAL uses the DLL and not the OSI presentation layer to transfer its APDUs. Therefore, there is no explicit presentation context available to the FAL. The FAL protocol may not be used concurrently with other application layer protocols between the same pair or set of data link service access points.

4.1.3.2 Fundamental concepts

The operation of time-critical real open systems is modelled in terms of interactions between time-critical APs. The FAL permits these APs to pass commands and data between them.

Cooperation between APs requires that they share sufficient information to interact and carry out processing activities in a coordinated manner. Their activities may be restricted to a single

fieldbus segment, or they may span multiple segments. The FAL has been designed using a modular architecture to support the messaging requirements of these applications.

Cooperation between APs also sometimes requires that they share a common sense of time. The FAL or the DLL may provide for the distribution of time to all devices. They may also define local device services that can be used by APs to access the distributed time.

The remainder of this subclause describes each of the modular components of the architecture and their relationships with each other. The components of the FAL are modelled as objects, each of which provides a set of FAL communication services for use by applications. The FAL objects and their relationships are described below. The detailed specifications of FAL objects and their services are provided in the following clauses of this standard. IEC 61158-6-21:2010 specifies the protocols necessary to convey these object services between applications.

4.1.3.3 Fieldbus application processes

4.1.3.3.1 Definition of the fieldbus AP

In the fieldbus environment, an application may be partitioned into a set of components and distributed across a number of devices on the network. Each of these components is referred to as a fieldbus AP. A fieldbus AP is a variation of an AP as defined in the ISO OSI Reference Model (ISO/IEC 7498-1). fieldbus APs may be addressed unambiguously by at least one individual DLL service access point address. Unambiguous addressing in this context means that no other AP may simultaneously be located at the same address. This definition does not prohibit an AP from being located at more than one individual or group data link service access point (DLSAP) address.

4.1.3.3.2 Communication services

Fieldbus APs communicate with each other using confirmed and unconfirmed services (ISO/IEC 10731). The services defined in this standard for the FAL specify the semantics of the services as seen by the requesting and responding APs. The syntax of the messages used to convey the service requests and responses is defined in IEC 61158-6-21:2010. The AP behavior associated with the services is specified by the AP.

Confirmed services are used to define request/response exchanges between APs.

On the other hand, unconfirmed services are used to define the unidirectional transfer of messages from one AP to one or more remote APs. 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.

4.1.3.3.3 AP interactions

4.1.3.3.3.1 General

In the fieldbus environment, APs may interact with other APs as necessary to achieve their functional objectives. No constraints are imposed by this standard on the organization of these interactions or the possible relationships that may exist between them.

For example, in the fieldbus environment, interactions may be based on request/response messages sent directly between APs, or on data/events sent by one AP for use by others. These two models of interaction between APs are referred to as client/server and publisher/subscriber interactions, respectively.

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 (for example.. client, server, peer, publisher, or subscriber) is defined as an attribute of the AREP.

4.1.3.3.3.2 Client/server interactions

Client/server interactions are characterized by bidirectional data flow between a client AP and one or more server APs. Figure 3 illustrates the interaction between a single client and a single server. In this type of interaction, the client may issue a confirmed or unconfirmed request to the server to perform some task. If the service is confirmed then the server will always return a response. If the service is unconfirmed, the server may return a response using an unconfirmed service defined for this purpose.

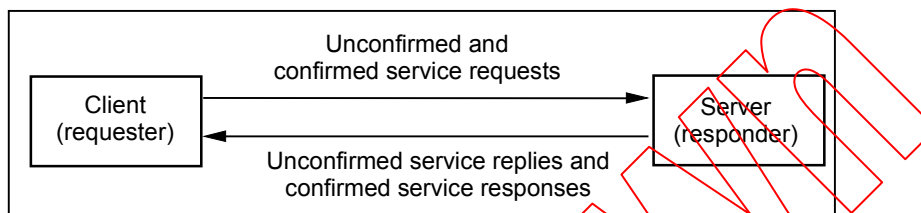


Figure 3 – Client/server interactions

4.1.3.3.3.3 Publisher/subscriber interactions

4.1.3.3.3.3.1 General

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 outside the scope of this standard. The Type 21 protocol supports only the “push” model.

4.1.3.3.3.3.2 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 are “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 others. First, a typical confirmed request/response exchange is performed between the 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 DLL 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 4 illustrates these concepts.

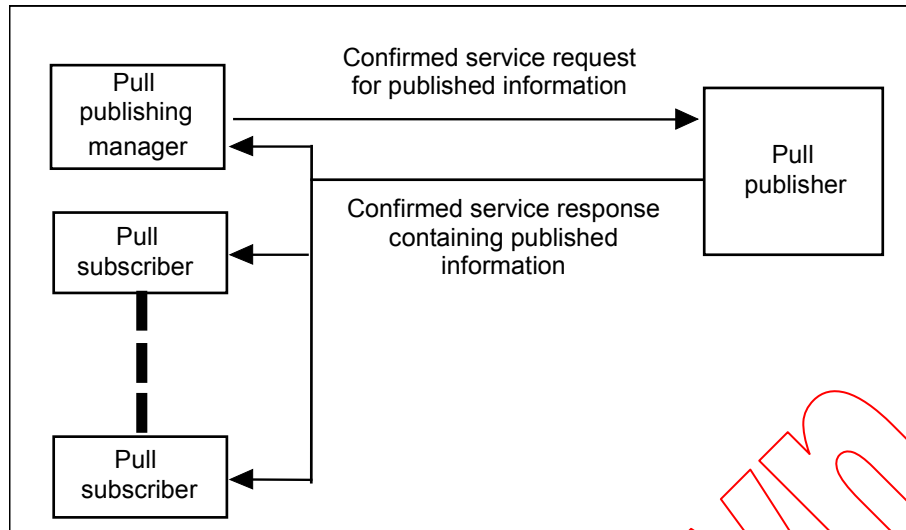


Figure 4 – Pull model interactions

4.1.3.3.3.3 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 in the request to subscribe to the publishing activity. 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 of 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 5 illustrates the concept of the push model.

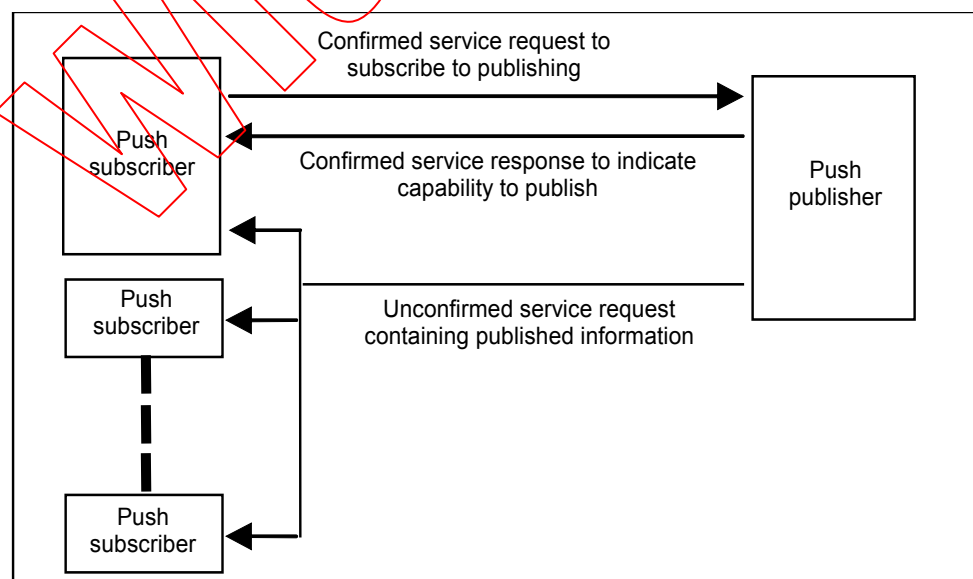


Figure 5 – Push model interactions

4.1.3.3.3.4 Timeliness of published information

To support the perishable nature of published information, the FAL may support four types of timeliness defined for publisher/subscriber interactions. Each makes it possible for subscribers of published data to determine if the data they are receiving is current or stale. These types are realized through mechanisms inside the DLL. Each is described briefly in Table 1.

Table 1 – Types of timeliness

Type	Description
Transparent	This type of timeliness allows the user application process to determine the timeliness quality of the data that it generates and have the timeliness quality accompany the information when it is transferred across the network. In this type, the network provides no computation or measurement of timeliness. It merely conveys the timeliness quality provided with the data by the user application process.
Residence	When the FAL submits data from the publishing AP to the DLL for transmission, the DLL starts a timer. If the timer expires before the data has been transmitted, the DLL marks the buffer as “not timely” and conveys this timeliness information with the data.
Synchronized	This type of timeliness requires the coordination of two pieces of published information: the data to be published and a special “sync mark.” When the sync mark is received from the network, a timer starts in each of the participating stations. Subsequently, when data are received for transmission by the DLL at the publishing station, or when the transmitted data are received from the network at a subscribing station, the DLL timeliness attribute for the data is set to TRUE. It remains TRUE until reception of the next sync mark or until the timer expires. Data received after the timer expires but before the next sync mark arrives do not cause the timeliness attribute to be reset to TRUE. It is only reset to TRUE if data are received within the time window after receipt of the sync mark. Data transmitted by the publisher station with the timeliness attribute set to FALSE maintains the setting of FALSE at each of the subscribers, regardless of their timer operation.
Update	This type of timeliness requires coordination of the same two pieces of published information defined for synchronized timeliness. In this type, the sync mark also starts a timer in each of the participating stations. Similar to synchronized timeliness, expiry of the timer always causes the timeliness attribute to be set to FALSE. Unlike synchronized timeliness, receipt of new data at any time (not just within the time window started with the receipt of a sync mark) causes the timeliness attribute to be set to TRUE.

4.1.3.3.4 AP structure

The internals of APs may be represented by one or more APOs and accessed through one or more AEs. AEs provide the communication capabilities of the AP. For each fieldbus AP, there is one and only one FAL AE. APOs are the network representation of application-specific capabilities (user application process objects) of an AP that are accessible through its FAL AE.

4.1.3.3.5 AP class

An AP class is a definition of the attributes and services of an AP. The standard class definition for APs is defined in this clause. User-defined classes may also be specified. Class identifiers, also described in this clause, are assigned from a set reserved for this purpose.

4.1.3.3.6 AP type

As described above in the previous subclauses, APs are defined by instantiating an AP class. Each AP definition is composed of the attributes and services selected for the AP from those defined by its AP class. In addition, an AP definition contains values for one or more of the attributes selected for it. When two APs share the same definition, that definition is referred to as an AP type. Thus, an AP type is a generic specification of an AP that may be used to define one or more APs.

4.1.3.4 Application process objects (APOs)

4.1.3.4.1 Definition

An APO 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 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 modelled as a network-accessible object contained in an AP or in another APO (APOs may contain other APOs). APOs provide the network definition for objects contained in 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 6, are provided by the FAL communications entity of the AP, known as the FAL applications entity (FAL AE).

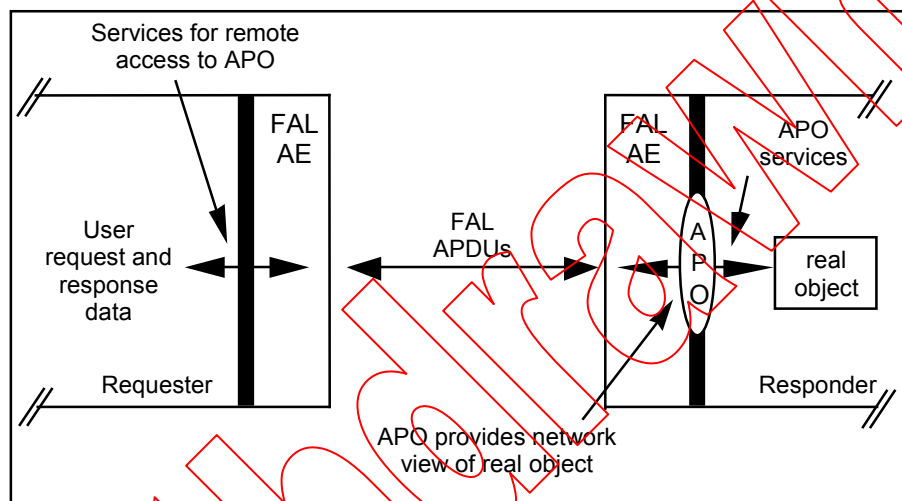


Figure 6 – APOs services conveyed by the FAL

In Figure 6, 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.

To support the publisher/subscriber model of interaction, information about the real object can be published through its APO. Remote APs acting as subscribers see the APO view of the published information instead of having to know any of the real object-specific details.

4.1.3.4.2 APO classes

An APO class is a generic specification for a set of APOs, each of which is described by the same set of attributes and accessed using the same set of services.

APO classes provide the mechanism for standardizing network-visible aspects of APs. Each standard APO class definition specifies a particular set of network-accessible AP attributes and services. IEC 61158-6-21:2010 specifies the syntax and the procedures used by the FAL protocol to provide remote access to the attributes and services of an APO class.

Standard APO classes are specified by this standard for the purpose of standardizing remote access to APs. User-defined classes may also be specified.

User-defined classes are defined as subclasses of standardized APO classes or of other user-defined classes. They may be defined by identifying new attributes or by indicating that optional attributes for the parent class are mandatory for the subclass. The conventions for

defining classes defined in this clause may be used for this purpose. The method for registering or otherwise making these new class definitions available for public use is outside the scope of this standard.

4.1.3.4.3 APOs as instances of APO classes

APO classes are defined in this standard using templates. These templates are used not only to define APO classes, but also to specify the instances of a class.

Each APO defined for an AP is an instance of an APO class. Each APO provides the network view of a real object contained in an AP. An APO is defined by:

- a) selecting the attributes from its APO class template that are to be accessible from the real object;
- b) assigning values to one or more attributes indicated as key in the template. Key attributes are used to identify the APO;
- c) assigning values to zero, one, or more non-key attributes for the APO. Non-key attributes are used to characterize the APO;
- d) selecting the services from the template that may be used by remote APs to access the real object.

Subclause 3.5.3 specifies the conventions for class templates. These conventions provide for the definition of mandatory, optional, and conditional attributes and services.

Mandatory attributes and services are required to be present in all APOs of the class. Optional attributes and services may be selected on an APO-by-APO basis for inclusion in an APO. Conditional attributes and services are defined with an accompanying constraint statement. Constraint statements specify the conditions under which the attribute is present in an APO.

4.1.3.4.4 APO types

APO types provide the mechanism for defining standard APOs.

As described above, APOs are defined by instantiating an APO class. Each APO definition is composed of the attributes and services selected for the APO from those defined by its APO class. In addition, an APO definition contains values for one or more of the attributes selected for it. When two APOs share the same definition except for the key attribute settings, that definition is referred to as an APO type. Thus, an APO type is a generic specification of an APO that may be used to define one or more APOs.

4.1.3.5 Application entities

4.1.3.5.1 Definition of FAL AE

An application entity provides the communication capabilities for a single AP. An FAL AE provides a set of services and the supporting protocols to enable communications between APs in a fieldbus environment. The services provided by FAL AEs are grouped into ASEs such that the FAL services provided to an AP are defined by the ASEs that its FAL AE contains. Figure 7 illustrates this concept.

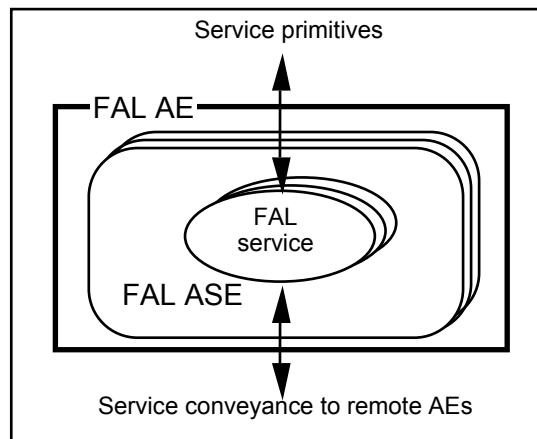


Figure 7 – Application entity structure

4.1.3.5.2 AE type

Application entities that provide the same set of ASEs are of the same AE-type. Two AEs that share a common set of ASEs are capable of communicating with each other.

4.1.3.6 Fieldbus ASEs

4.1.3.6.1 General

An ASE, as defined in ISO/IEC 9545, is a set of application functions that provide a capability for the interworking of application entity-invocations for a specific purpose. ASEs provide a set of services for conveying requests and responses to and from application processes and their objects. AEs, as defined above, are represented by a collection of ASE invocations within the AE.

4.1.3.6.2 FAL services

FAL services convey functional requests/responses between APs. Each FAL service is defined to convey requests and responses for access to a real object modeled as an FAL accessible object.

The FAL defines both confirmed and unconfirmed services. Confirmed service requests are sent to the AP containing the real object. An invocation of a confirmed service request may be identified by a user-supplied invoke ID. This invoke ID is returned in the response by the AP containing the real object. When present, it is used by the requesting AP and its FAL AE to associate the response with the appropriate request.

Unconfirmed services may be sent from the AP containing the real object to send information about the object. They also may be sent to the AP containing the real object to access the real object. Both types of unconfirmed services may be defined for the FAL.

4.1.3.6.3 Definition of FAL ASEs

4.1.3.6.3.1 General

4.1.3.6.3.2 Object-management ASE

A special object-management ASE may be specified for the FAL to provide services for the management of objects. Its services are used to access object attributes, and create and delete object instances. These services are used to manage network-visible AP objects accessed through the FAL. The specific operational services that apply to each object type are

specified in the definition of the ASE for the object type. Figure 8 illustrates the integration of management and operational services for an object within an AP.

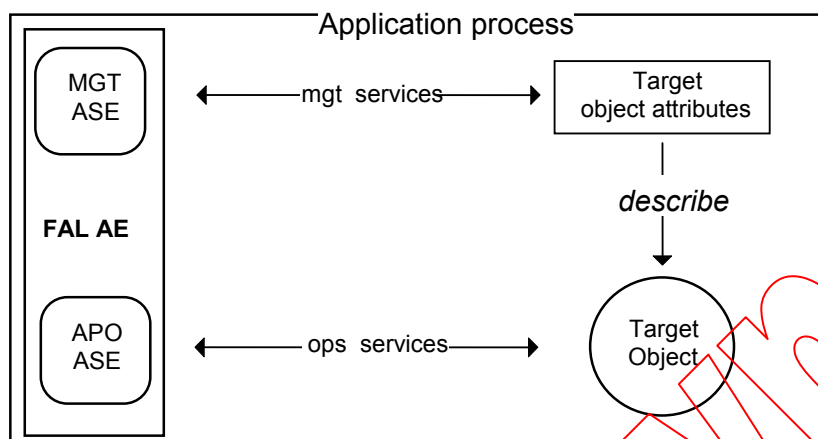


Figure 8 – FAL management of objects

4.1.3.6.3.3 AP ASE

An AP ASE may be specified for the identification and control of FAL APs. The attributes defined by the AP ASE give details about the AP's creator, and list its contents and capabilities.

4.1.3.6.3.4 APO ASEs

The FAL specifies a set of ASEs with services defined for accessing the APOs of an AP. The APO ASEs defined for the FAL are defined by each communication model.

4.1.3.6.3.5 AR ASE

An AR ASE is specified to establish and maintain Ars, which are used to convey FAL APDUs between or among APs. ARs represent application layer communication channels between APs. AR ASEs are responsible for providing services at the endpoints of ARs. AR ASE services may be defined for establishing, terminating, and aborting ARs. They may also be defined for conveying APDUs for the AE, and for informing the user of the local status of the AR. In addition, local services may be defined for accessing certain aspects of AR endpoints.

4.1.3.6.4 FAL service conveyance

FAL APO ASEs provide services to convey requests and responses between service users and real objects.

To convey service requests and responses, there are three types of activities defined for the sending user, and three corresponding types defined for the receiving user. At the sending user, FAL APO ASEs accept service requests and responses to be conveyed. They also select the type of FAL APDU that will be used to convey the request or response, and encode the service parameters in the body portion. Then, they submit the encoded APDU body to the AR ASE for conveyance.

At the receiving user, FAL APO ASEs receive encoded APDU bodies from the AR ASE. They decode the APDU bodies and extract the service parameters conveyed by them. Then, they deliver the service request or response to the user. Figure 9 illustrates these concepts.

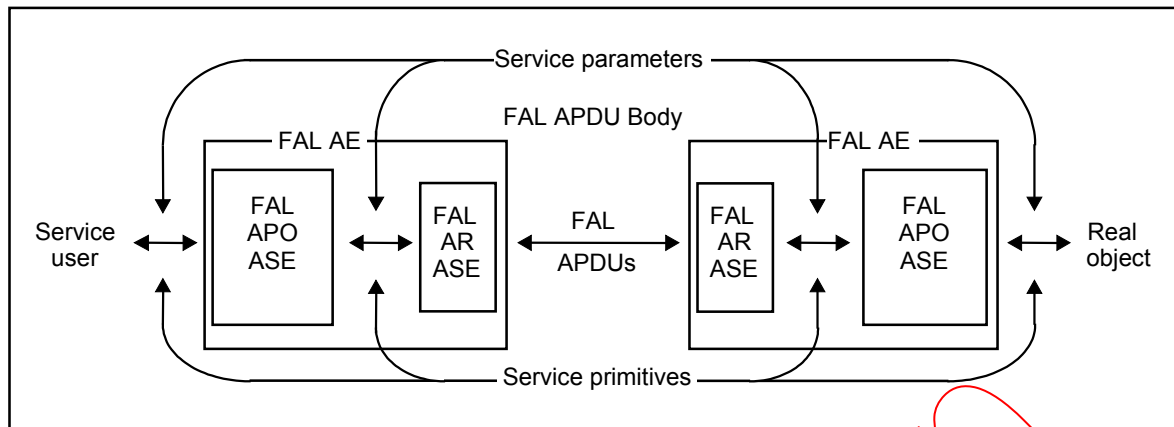


Figure 9 – ASE service conveyance

4.1.3.6.5 FAL presentation context

The presentation context in the OSI environment is used to distinguish the APDUs of one ASE from another, and to identify the transfer syntax rules used to encode each APDU. However, the fieldbus communications architecture does not include a presentation layer. Therefore, an alternate mechanism is provided for the FAL by each of the specific types of communication models.

4.1.3.7 Application relationships

4.1.3.7.1 Definition of AR

ARs represent communication channels between APs. They define how information is communicated between APs. Each AR is characterized by how it conveys ASE service requests and responses from one AP to another. These characteristics are described below.

4.1.3.7.2 AR-endpoints

ARs are defined as a set of cooperating APs. The AR ASE in each AP manages an endpoint of the AR, and maintains its local context. The local context of an AR endpoint is used by the AR ASE to control the conveyance of APDUs on the AR.

4.1.3.7.3 AR-endpoint classes

ARs are composed of a set of endpoints of compatible classes. AR endpoint classes are used to represent AR endpoints that convey APDUs in the same way. Through the standardization of endpoint classes, ARs for different models of interaction can be defined.

4.1.3.7.4 AR cardinality

ARs characterize communications between APs. One of the characteristics of an AR is the number of AR endpoints in the AR. ARs that convey services between two APs have a cardinality of 1:1. Those that convey services from one AP to a number of APs have a cardinality of 1:many. Those that convey services to or from multiple APs have a cardinality of many:many.

4.1.3.7.5 Accessing objects through ARs

ARs provide access to APs and the objects within them via the services of one or more ASEs. Therefore, one characteristic is the set of ASE services that may be conveyed to and from these objects by the AR. The list of services that can be conveyed by the AR is selected from those defined for the AE.

4.1.3.7.6 AR conveyance paths

ARs are modeled as one or two conveyance paths between AR endpoints. Each path conveys APDUs in one direction between one or more AR endpoints. Each receiving AR endpoint for a conveyance path receives all APDUs transmitted on the AR by the sending AR endpoint.

4.1.3.7.7 AREP roles

As APs interact with each other through endpoints, a basic determinant of their compatibility is the role that they play in the AR. The role defines how an AREP interacts with other AREPs in the AR.

For example, an AREP may operate as a client, a server, a publisher, or a subscriber. When an AREP interacts with another AREP on a single AR as both a client and a server, it is defined to have the role of “peer.”

Certain roles may be capable of initiating service requests, while others may be capable only of responding to service requests. This part of the definition of a role identifies the requirement for an AR to be capable of conveying requests in either direction, or in only one direction.

4.1.3.7.8 AREP buffers and queues

AREPs may be modelled as a queue or as a buffer. APDUs transferred over a queued AREP are delivered in the order received for conveyance. The transfer of APDUs over a buffered AREP is different. In this case, an APDU to be conveyed by the AR ASE is placed in a buffer for transfer. When the DLL gains access to the network, it transmits the contents of the buffer.

When the AR ASE receives another conveyance request, it replaces the previous contents of the buffer regardless of whether they were transmitted. Once an APDU is written into a buffer for transfer, it is preserved in the buffer until it is overwritten by the next APDU to be transmitted. While in the buffer, an APDU may be read more than once without deleting it from the buffer or changing its contents.

The operation is similar at the receiving end. The receiving endpoint places a received APDU into a buffer for access by the AR ASE. When a subsequent APDU is received, it overwrites the previous APDU in the buffer regardless of whether or not it was read. Reading the APDU from the buffer is not destructive; it does not destroy or change the contents of the buffer, allowing the contents to be read from the buffer one or more times.

4.1.3.7.9 User-triggered and scheduled conveyance

Another characteristic of an AREP is the time frame in which it conveys service requests and responses. User-triggered AREPs convey requests and responses immediately upon submission by the user. This is asynchronous with respect to network operation.

A scheduled AREP conveys requests and responses at predefined intervals, regardless of when they are received for transfer. A scheduled AREP may be capable of indicating when transferred data was submitted late for transmission, or when it was submitted on time but transmitted late.

4.1.3.7.10 AREP timeliness

AREPs convey APDUs between applications using the services of the DLL. When the timeliness capabilities are defined for an AREP and supported by the DLL, the AREP forwards the timeliness indicators provided by the DLL. These indicators make it possible for subscribers of published data to determine if the data they are receiving is current or stale.

To support these types of timeliness, the publishing AREP establishes a publisher data link connection reflecting the type of timeliness configured for it by management. After establishing the connection, the AREP receives user data and submits it to the DLL for transmission where the timeliness procedures are performed. When the DLL transmits the data, it includes the current timeliness status with the data.

At the subscriber AREP, a data line connection is opened to receive published data that reflects the type of timeliness configured for it by management. The DLL computes the timeliness of received data and then delivers it to the AREP. The data are then delivered to the user AP through the appropriate ASE.

4.1.3.7.11 Definition and creation of AREPs

AREP definitions specify instances of AREP classes. AREPs may be predefined or they may be defined using a “create” service if their AE supports this capability.

AREPs may be pre-defined and pre-established, or they may be pre-defined and established dynamically. Figure 10 depicts these two cases. AREPs also may require both dynamic definition and establishment or they may be defined dynamically in such a way that they may be used without any establishment, *i.e.*, they are defined in an established state.

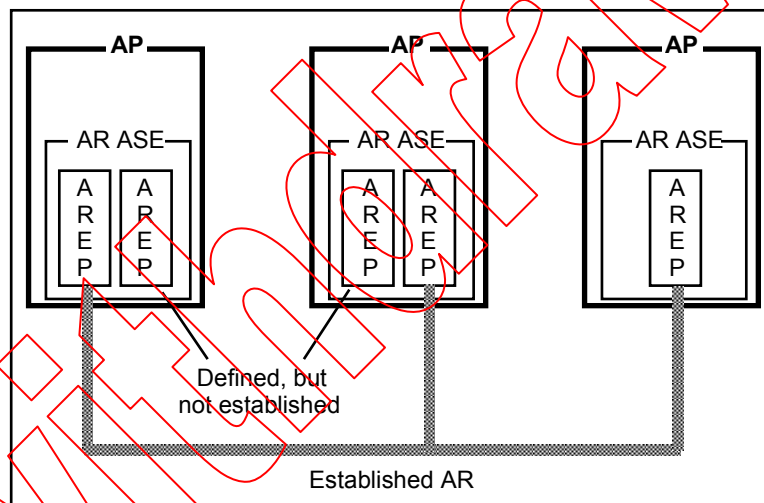


Figure 10 – Defined and established AREPs

4.1.3.7.12 AR establishment and termination

ARs may be established either before the operational phase of the AP or during its operation. When established during the operation of an AP, the AR is established through the exchange of AR APDUs.

Once an AR has been established, an AR may be terminated gracefully or it may be aborted, depending on the capabilities of the AR.

4.1.4 Fieldbus application layer naming and addressing

4.1.4.1 General

This clause refines the principles defined in ISO 7498-3 that involve the identification (naming) and location (addressing) of APOs referenced through the FAL.

This clause also defines how names and numeric identifiers are used to identify APOs accessible through the FAL. In addition, this clause indicates how addresses from underlying layers are used to locate APs in the fieldbus environment.

4.1.4.2 Identifying objects accessed through the FAL

4.1.4.2.1 General

APOs accessed through the FAL are identified independent of their location. That is, if the location of the AP containing the APO changes, the APO may still be referenced using the same set of identifiers.

Identifiers for APs and APOs in the FAL are defined as key attributes in the class definitions for APOs. Two types of key attribute are commonly used in these APO definitions: names and numeric identifiers.

4.1.4.2.2 Names

Names are string-oriented identifiers. They are defined to permit APs and APOs to be named in the system where they are used. Therefore, although the scope of an APO name is specific to the AP in which it resides, the assignment of the name is administered in the system in which it is configured.

Names may be descriptive, although this is not mandatory. Descriptive names make it possible to provide meaningful information about the object they name, such as its use.

Names may also be coded. Coded names make it possible to identify an object using a short, compressed form of a name. They are typically simpler to transfer and process, but more difficult to understand than descriptive names.

4.1.4.2.3 Numeric identifiers

Numeric identifiers are identifiers whose values are numbers. They are designed for efficient use in the fieldbus system, and may be assigned to APOs by their AP for efficient access.

4.1.4.3 Addressing APs accessed through the FAL

Fieldbus addresses represent the network locations of APs. Addresses relevant to the FAL are the addresses of the underlying layers that are used to locate the AREPs of an AP.

4.1.5 Architecture summary

This clause presents a summary of the FAL architecture. Figure 11 illustrates the major components and how they relate to each other.

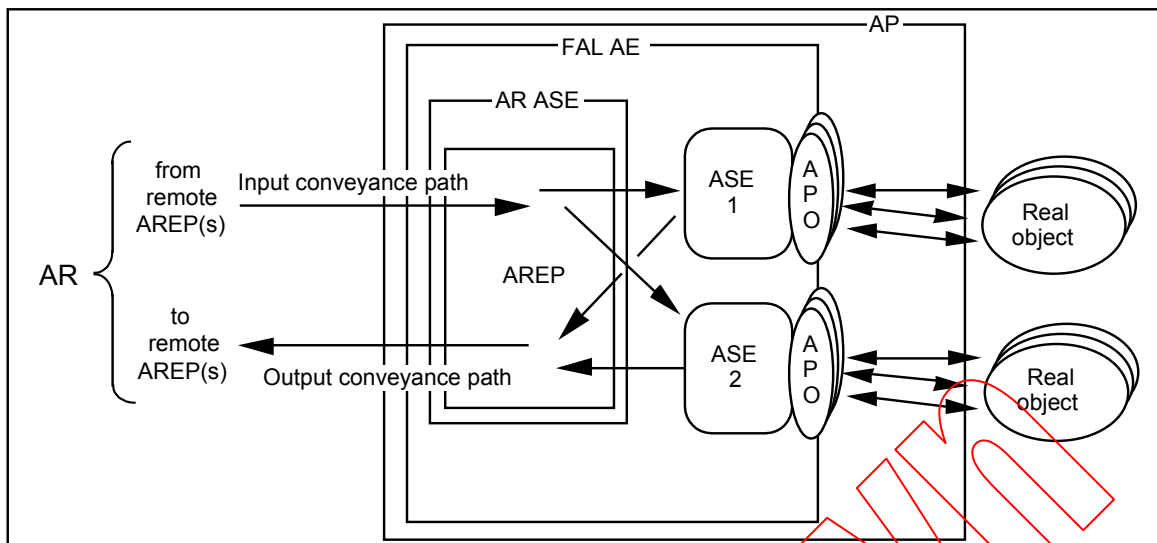


Figure 11 – FAL architectural components

Figure 11 shows an AP that communicates through the FAL AE. The AP represents its internal real objects as APOs for remote access to them. Two ASEs that provide remote access services to their related APOs are shown. The AR ASE contains a single AREP that conveys service requests and responses for the ASEs to one or more remote AREPs located in remote APs.

4.1.6 FAL service procedures

4.1.6.1 FAL confirmed service procedures

The requesting user invokes a confirmed-service request primitive of its FAL. The appropriate FAL ASE builds the related confirmed-service-request APDU body and conveys it on the specified AR.

Upon receipt of the confirmed-service request APDU body, the responding ASE decodes it. If a protocol error did not occur, the ASE delivers a confirmed-service indication primitive to its user.

If the responding user is able to process the request successfully, the user returns a confirmed-service response (+) primitive.

If the responding user is unable to process the request successfully, the service fails and the user issues a confirmed-service response (-) primitive indicating the reason.

The responding ASE builds a confirmed-service-response APDU body for a confirmed-service response (+) primitive or a confirmed-service-error APDU body for a confirmed-service response (-) primitive and conveys it on the specified AR.

Upon receipt of the returned APDU body, the initiating ASE delivers a confirmed-service confirmation primitive to its user that specifies success or failure, and the reason for failure if a failure occurred.

4.1.6.2 FAL unconfirmed service procedures

The requesting user invokes an unconfirmed-service request primitive from its FAL AE. The appropriate FAL ASE builds the related unconfirmed-service request APDU body and conveys it on the specified AR.

Upon receipt of the unconfirmed-request APDU body, the receiving ASE(s) participating in the AR delivers the appropriate unconfirmed-service indication primitive to its user. Timeliness parameters are included in the indication primitive if the AR that conveyed the APDU body supports timeliness.

4.1.7 Common FAL attributes

In the specifications of the FAL classes that follow, many classes use the following common attributes. Therefore, these attributes are defined once here instead of with the other attributes for each individual class, except for the data type class.

ATTRIBUTES:

- | | | | |
|---|-----|----------------|--------------------|
| 1 | (o) | Key attribute: | Numeric identifier |
| 2 | (o) | Key attribute: | Name |
| 3 | (o) | Attribute: | User description |
| 4 | (o) | Attribute: | Object revision |

Numeric identifier

optional key attribute specifies the numeric ID of the object. It is used as a shorthand reference by the FAL protocol to identify the object. There are three possibilities for identification purposes: numeric identifier, name, or both. This attribute is required for the data type model.

Name

optional key attribute specifies the name of the object. There are three possibilities for identification purposes: numeric identifier, name, or both.

User description

optional attribute contains user-defined descriptive information about the object.

Object revision

optional attribute specifies the revision level of the object. It is a structured attribute composed of major and minor revision numbers. If object revision is supported, it contains both a major revision and a minor revision with a value in the range 0 to 15 for each. The major and minor revision fields are used as follows:

- a) the major revision field contains the major revision value for the object. A change to the major revision indicates that interoperability is affected by the change;
- b) the minor revision field contains the minor revision value for the object. A change to the minor revision indicates that interoperability was not affected by the change, i.e., users of the object will continue to be capable of interoperating with the object if its minor revision is changed, provided that the major revision remains the same.

4.1.8 Common FAL service parameters

In the specifications of the FAL services that follow, many services use the following parameters. Therefore, they are defined once here instead of with the other parameters for each of the services.

AREP

parameter contains sufficient information to identify the AREP locally so it can be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship. When an AREP supports multiple contexts (established using the initiate service) at the same time, the AREP parameter is extended to identify the context as well as the AREP

Invoke ID

parameter identifies this invocation of the service. It is used to associate service requests with responses. Therefore, no two outstanding service invocations may be identified by the same invoke ID value

FAL ASE/FAL class

parameter specifies the FAL ASE (e.g., AP, AR, variable, data type, event, function-invocation, load-region) and the FAL class of the ASE (for example. AREP, variable-list, notifier, action)

Numeric ID

parameter is the numeric identifier of the object

Error info

parameter provides error information for service errors. It is returned in confirmed service response (-) primitives. It is composed of the following elements:

- a) **Error class.** This parameter indicates the general class of error. Valid values are specified in the definition of the error code parameter below;
- b) **Error code.** This parameter identifies the specific service error;
- c) **Additional code.** If an error is encountered when processing the request, this optional parameter identifies the error specific to the object being accessed. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive;
- d) **Additional detail.** This optional parameter contains user data that accompanies a negative response. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive.

4.1.9 APDU size

The APDU size is dependent on the communication model.

4.2 Type specific concepts

The industrial automation and process control system consists of primary automation devices (e.g., sensors, actuators, local display devices, annunciators, programmable logic controllers, small single loop controllers, and standalone field controls) with control and monitoring equipment.

Data transfer between these devices is performed by peer-to-peer or multicast communication.

Figure 12 illustrates the interaction between Type 21 FAL and the DLL. Type 21 supports cyclic and acyclic data transfer for its own application processes. Type 21 can also be used in parallel with TCP/IP or UDP communication. The use of other standard communication protocols is beyond this standard.

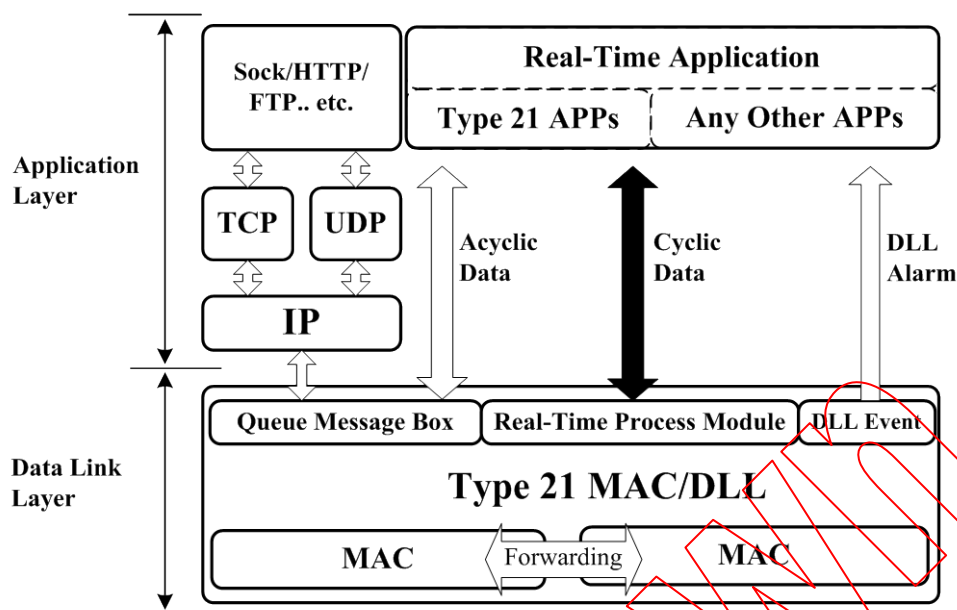


Figure 12 – Interaction between FAL and DLL

Type 21 supports the publisher-subscriber communication model for cyclic data sharing. Figure 13 illustrates this model. The publisher periodically multicasts preconfigured data, and subscribers receive the data. This cyclic data sharing is the most widely used model in industrial applications.

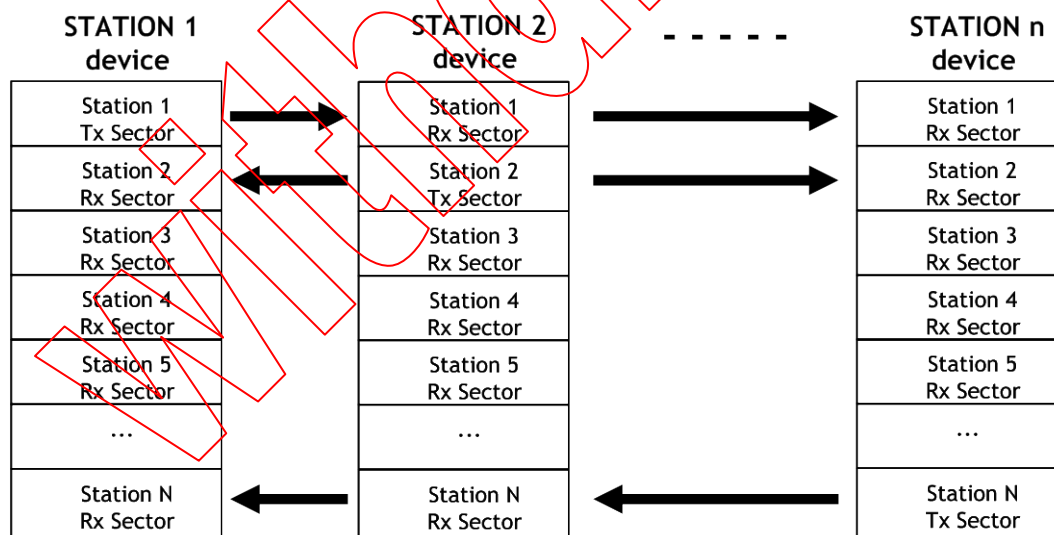


Figure 13 – Publisher-subscriber communication model

Type 21 supports the client-server communication model for event-triggered data transfer. Figure 14 illustrates this model. The client requests data as dictated by internal or external events. The server replies with the data. This can be used for event-triggered or user-triggered application processes.

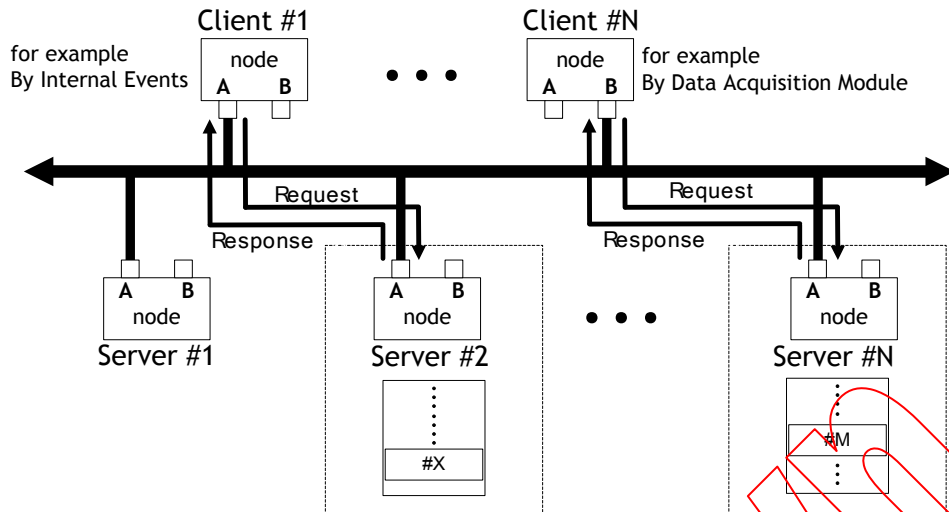


Figure 14 – Client-server communication model

4.2.1 Node, AP, and object dictionary

Each node hosts exactly one AP. All APOs for this AP are collected in the object dictionary (OD). Figure 15 illustrates the object model.

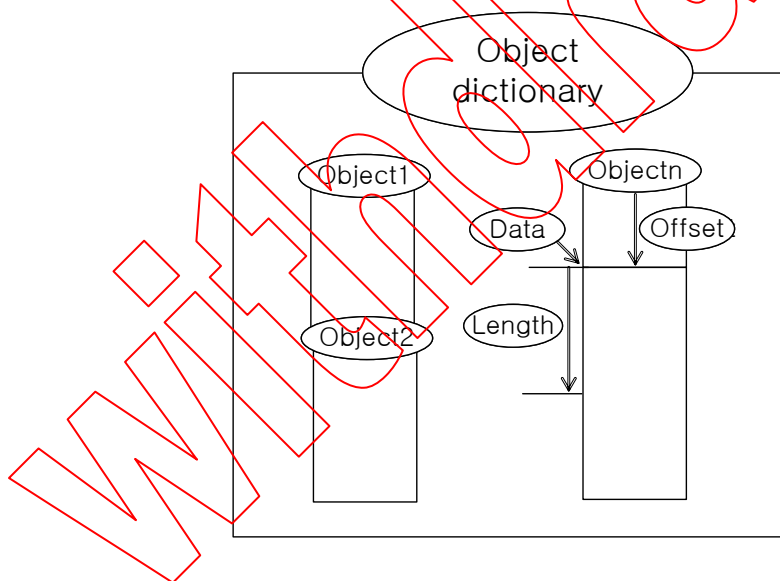


Figure 15 – Object model

The overall structure of the OD is as described in Table 2.

Table 2 – Overall structure of the OD

Area	Contents
Data type area	Definition of data types
Communication profile area	Contains communication-specific parameters for the Type 21 network. These entries are common to all devices.
Manufacturer-specific area	Definition of manufacturer-specific variables
Device profile area	Definition of the variables defined in a device profile (outside the scope of this document)
Reserved area	Reserved for future use

4.2.2 APO ASEs

The FAL ASEs of a Type 21 application and their interrelationships are described in Figure 16.

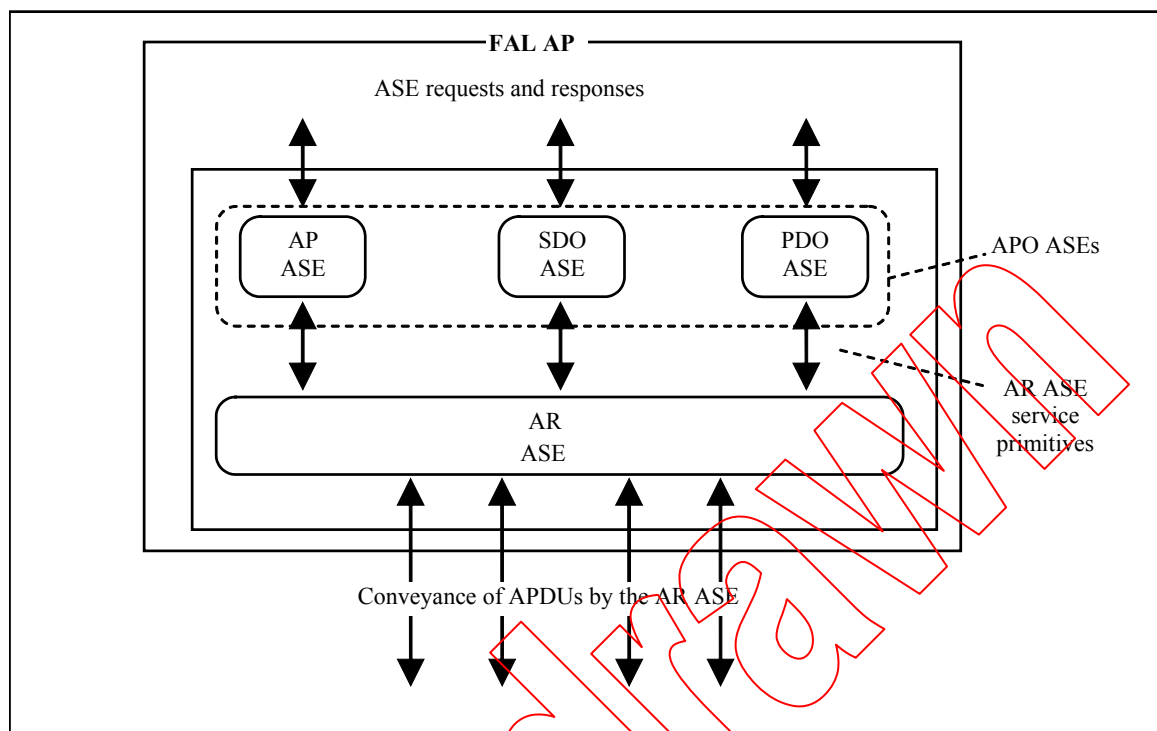


Figure 16 – ASEs of a Type 21 application

5 Data type ASE

5.1 General

5.1.1 Overview

The fieldbus data types specify the machine-independent syntax for application data conveyed by FAL services. The FAL supports the definition and transfer of both basic and constructed data types. The encoding rules for the data types specified in this clause are given in IEC 61158-6-21:2010.

Basic types are atomic types that cannot be decomposed. Constructed types are types composed of basic types and other constructed types. This standard does not constrain their complexity or depth of nesting.

Data types are defined as instances of the data type class, as shown in Figure 17. Only a subset of the data types defined in this clause are shown in this figure. Defining new types is accomplished by providing a numeric ID and supplying values for the attributes defined for the data type class.

The data type definitions shown in Figure 17 are represented as a class/format/instance structure beginning with the "Data type" data type class.

The basic data classes are used to define fixed-length and bit string data types. Standard types taken from ISO/IEC 8824 are referred to as simple data types. Other standard basic data types are defined specifically for fieldbus applications, and are referred to as specific types.

The constructed types specified in this standard are strings, arrays, and structures. There are no standard types defined for arrays or structures.

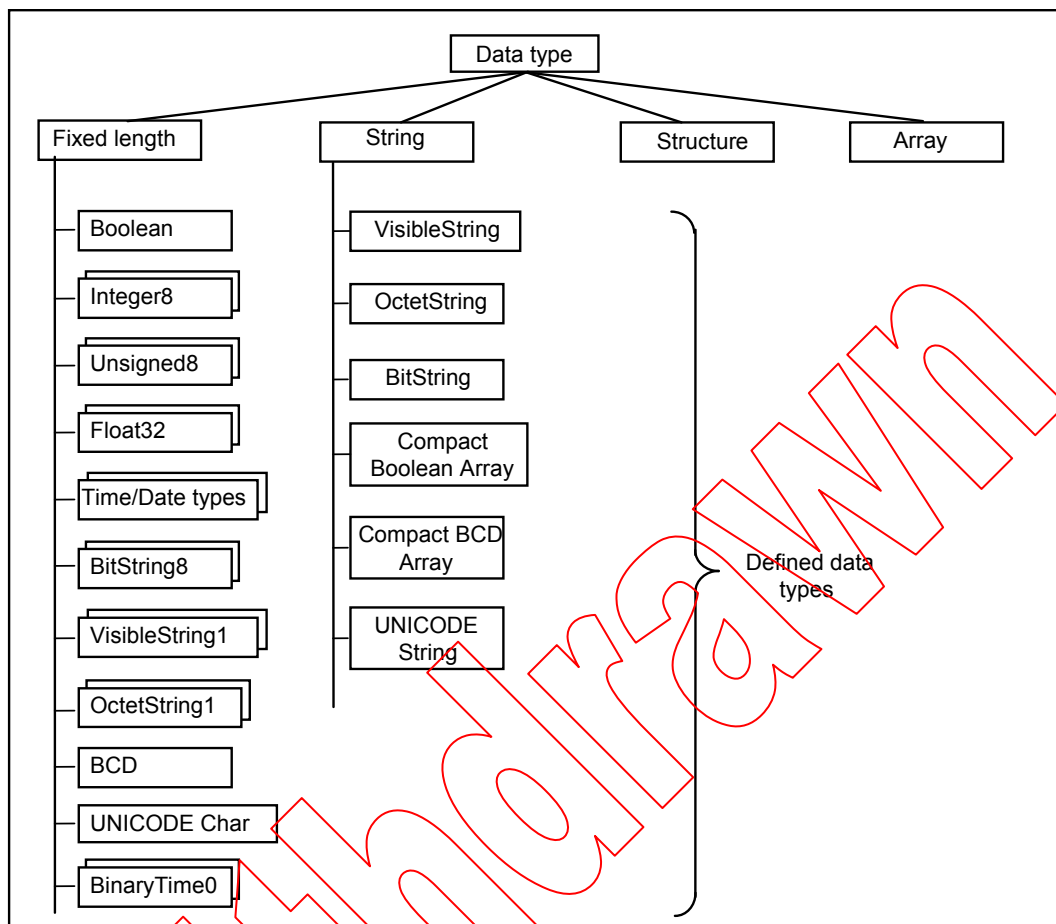


Figure 17 – Data type class hierarchy example

5.1.2 Basic type overview

Most basic types are defined from a set of ISO/IEC 8824 types (simple types). Some ISO/IEC 8824 types have been extended for fieldbus specific use (specific types).

Simple types are ISO/IEC 8824 universal types. They are defined in this standard to provide them with fieldbus class identifiers.

Specific types are basic types defined specifically for use in the fieldbus environment. They are defined as simple class subtypes.

Basic types have a constant length. Two variations are defined, one for defining data types whose length is an integral number of octets, and one for defining data types whose length is an arbitrary number of bits.

NOTE Boolean, Integer, OctetString, VisibleString, and UniversalTime are defined in this standard for the purpose of assigning fieldbus class identifiers to them. This standard does not change their definitions as specified in ISO/IEC 8824.

5.1.3 Fixed-length type overview

The length of fixed-length types is an integral number of octets.

5.1.4 Constructed type overview

5.1.4.1 Strings

A string is composed of an ordered set, variable in number, of homogeneously typed fixed-length elements.

5.1.4.2 Arrays

An array is composed of an ordered set of homogeneously typed elements. This standard places no restriction on the data type of array elements, but it does require that each element be of the same type. Once defined, the number of elements in an array may not be changed.

5.1.4.3 Structures

A structure is made of an ordered set of heterogeneously typed elements called fields. Like arrays, this standard does not restrict the data type of fields. However, the fields within a structure do not have to be of the same type.

5.1.4.4 Nesting level

This standard permits arrays and structures to contain other arrays and structures. It places no restriction on the number of nesting levels allowed. However, the FAL services defined to access data provide for partial access to the level negotiated by the Initiate service. The default number of levels for partial access is one.

When an array or structure contains constructed elements, access to a single element in its entirety is always provided. Access to sub-elements of the constructed element is also provided, but only when explicitly negotiated during AR establishment or when explicitly preconfigured on pre-established ARs.

NOTE For example, suppose that a data type named "employee" is defined to contain the structure "employee name," and "employee name" is defined to contain "last name" and "first name." To access the "employee" structure, the FAL permits independent access to the entire structure and to the first level field "employee name." Without explicitly negotiating partial access to more than one level, independent access to "last name" or "first name" would not be possible; their values could only be accessed together as a unit through access to "employee" or "employee name."

5.1.5 Specification of user-defined data types

Users may find it necessary to define custom data types for their own applications. User-defined types are supported by this standard as instances of data type classes.

User-defined types are specified in the same manner that all FAL objects are specified. They are defined by providing values for the attributes specified for their class.

5.1.6 Transfer of user data

User data are transferred between applications by the FAL protocol. All encoding and decoding is performed by the FAL-user.

The rules for encoding user data in FAL protocol data units depend on the data type. These are defined in IEC 61158-6-21:2010. User-defined data types for which there are no encoding rules are transferred as variable-length sequences of octets. The format of the data within the octet string is defined by the user.

5.2 Formal definition of data type objects

5.2.1 Data type class

5.2.1.1 Template

The data type class specifies the root of the data type class tree. Its parent class “TOP” indicates the top of the FAL class tree.

FAL ASE:		DATA TYPE ASE
CLASS:		DATA TYPE
CLASS ID:		5 (FIXED-LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)
PARENT CLASS:		TOP
ATTRIBUTES:		
1	(o) Key attribute:	Data type numeric identifier
2	(o) Key attribute:	Data type name
3	(m) Attribute:	Format (FIXED-LENGTH, STRING, STRUCTURE, ARRAY)
4	(c) Constraint:	Format = FIXED-LENGTH STRING
4.1	(m) Attribute:	Octet length
5	(c) Constraint:	Format = STRUCTURE
5.1	(m) Attribute:	Number of fields
5.2	(m) Attribute:	List of fields
5.2.1	(o) Attribute:	Field name
5.2.2	(m) Attribute:	Field data type
6	(c) Constraint:	Format = ARRAY
6.1	(m) Attribute:	Number of array elements
6.2	(m) Attribute :	Array element data type

5.2.1.2 Attributes

Data type numeric identifier

optional attribute identifies the numeric identifier of the related data type

Data type name

optional attribute identifies the name of the related data type

Format

attribute identifies the data type as a fixed-length, string, array, or data structure

Octet length

conditional attribute defines the representation of the dimensions of the associated type object. It is present when the value of the format attribute is “FIXED-LENGTH” or “STRING.” For FIXED-LENGTH data types, it represents the length in octets. For STRING data types, it represents the length in octets for a single element of a string

Number of fields

conditional attribute defines the number of fields in a structure. It is present when the value of the format attribute is “STRUCTURE.”

List of fields

conditional attribute is an ordered list of fields contained in the structure. Each field is specified by its number and its type. Fields are numbered sequentially from 0 (zero) in the order in which they occur. Partial access to fields within a structure is supported by identifying the field by number. This attribute is present when the value of the format attribute is “STRUCTURE.”

Field name

conditional, optional attribute specifies the name of the field. It may be present when the value of the format attribute is “STRUCTURE.”

Field data type

conditional attribute specifies the data type of the field. It is present when the value of the format attribute is "STRUCTURE." This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric ID, or by embedding a constructed data type definition here. When embedding a description, the embedded-data type description shown below is used

Number of array elements

conditional attribute defines the number of elements for the array type. Array elements are indexed from "0" through "n-1" for an array of "n" elements. This attribute is present when the value of the format attribute is "ARRAY."

Array element data type

conditional attribute specifies the data type for the elements of an array. All elements of the array have the same data type. It is present when the value of the format attribute is "ARRAY." This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric ID, or by embedding a constructed data type definition here. When embedding a description, the embedded-data type description shown below is used

Embedded-data type description

attribute is used to define embedded data types recursively in a structure or array. The template below defines its contents. The attributes shown in the template are defined above in the data type class, except for the embedded-data type attribute, which is a recursive reference to this attribute. It is used to define nested elements

ATTRIBUTES:

1	(m)	Attribute:	Format(FIXED-LENGTH, STRING, STRUCTURE, ARRAY)
2	(c)	Constraint:	Format = FIXED-LENGTH STRING
2.1	(m)	Attribute:	Data type numeric ID value
2.2	(m)	Attribute:	Octet length
3	(c)	Constraint:	Format = STRUCTURE
3.1	(m)	Attribute:	Number of fields
3.2	(m)	Attribute:	List of fields
3.2.1	(m)	Attribute:	Embedded data type description
4	(c)	Constraint:	Format = ARRAY
4.1	(m)	Attribute:	Number of array elements
4.2	(m)	Attribute:	Embedded data type description

5.3 FAL defined data types**5.3.1 Fixed-length types****5.3.1.1 Boolean types****CLASS:****Data type****ATTRIBUTES:**

1	Data type numeric identifier	=	1
2	Data type name	=	Boolean
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	1

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

5.3.1.2 Date/time types

5.3.1.2.1 TimeOfDay

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	12
2	Data type name	=	TimeOfDay
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	6

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type that gives the time after midnight in milliseconds. The second element is an Unsigned16 data type that gives the date as a count of the days from 1984-01-01.

5.3.1.2.2 TimeDifference

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	13
2	Data type name	=	TimeDifference
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	6

This data type is composed of two elements of unsigned values that express a length of time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The second element is an Unsigned16 data type that provides the number of days.

5.3.1.3 Numeric types

5.3.1.3.1 Real32

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	8
2	Data type name	=	Real32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

This type has a length of four octets. The format for Real32 is that defined by IEC 60559 as single precision.

5.3.1.3.2 Real64

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	17
2	Data type name	=	Real64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

This type has a length of eight octets. The format for Real64 is that defined by IEC 60559 as double precision.

5.3.1.3.3 Integer types

5.3.1.3.3.1 Integer8

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	2
2	Data type name	=	Integer8
3	Format	=	FIXED-LENGTH
4.1	Octet length	=	1

This integer type is a two's complement binary number with a length of one octet.

5.3.1.3.3.2 Integer16

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	3
2	Data type name	=	Integer16
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	2

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

5.3.1.3.3.3 Integer32

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	4
2	Data type name	=	Integer32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

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

5.3.1.3.3.4 Integer64

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	21
2	Data type name	=	Integer64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

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

5.3.1.3.4 Unsigned types

5.3.1.3.4.1 Unsigned8

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	5
2	Data type name	=	Unsigned8
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	1

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

5.3.1.3.4.2 Unsigned16

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	6
2	Data type name	=	Unsigned16
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	2

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

5.3.1.3.4.3 Unsigned32

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	7
2	Data type name	=	Unsigned32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

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

5.3.1.3.4.4 Unsigned64

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	27
2	Data type name	=	Unsigned64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

This type is a binary number. The most significant bit of the most significant octet is always the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of eight octets.

5.3.2 String types

5.3.2.1 OctetString

CLASS:		Data type	
ATTRIBUTES:			
1	Data type numeric identifier	=	10
2	Data type name	=	OctetString
3	Format	=	STRING
3.1	Octet length	=	1 to n

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet one of the sequence is referred to as the first octet. The order of transmission is defined in IEC 61158-6-21:2010.