IEC 61158-5-21

Edition 1.0 2010-08

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 – Eléments de Type 21

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It also gives information on projects, replaced and withdrawn publications.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.

IEC 61158-5-21

Edition 1.0 2010-08

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 – Eléments de Type 21

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XB
CODE PRIX

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-88912-865-5

CONTENTS

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.

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 61158-2:2010[1], *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-3-21:2010[1], *Industrial communication networks – Fieldbus specifications – Part 3-21: Data-link layer service definition – Type 21 elements*

IEC 61158-4-21:2010[1], *Industrial communication networks – Fieldbus specifications – Part 4-21: Data-link layer protocol specification – Type 21 elements*

IEC 61158-6-21:2010[1], *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) DLPDU

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 than 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.

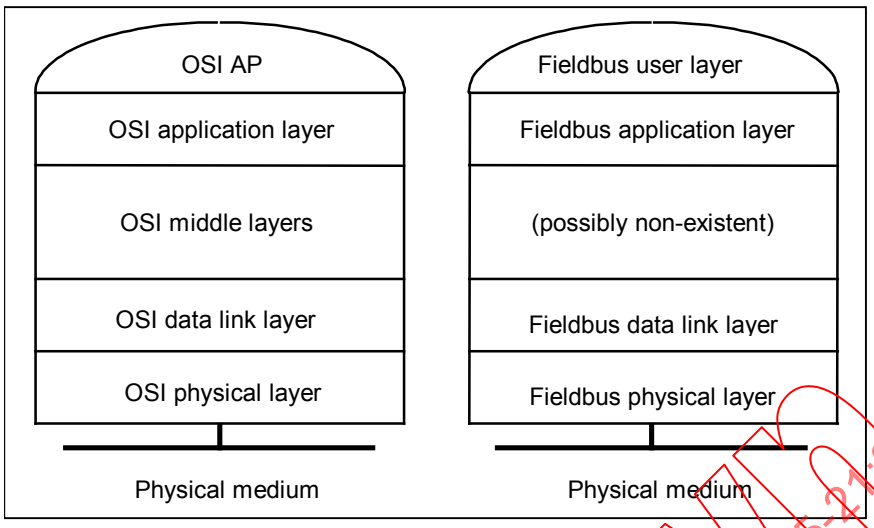


Figure 1 – Relationship to the OSI Basic Reference Model

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.

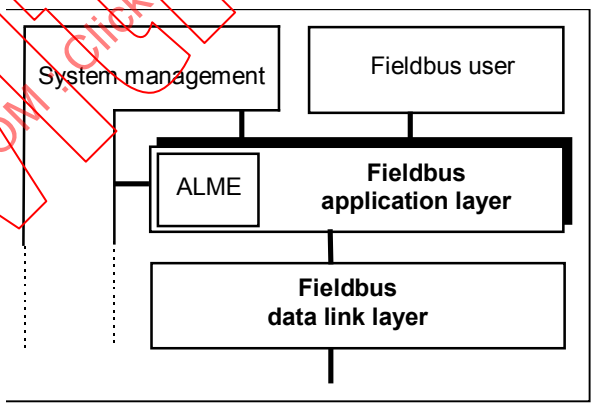


Figure 2 – Architectural positioning of the fieldbus application layer

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.

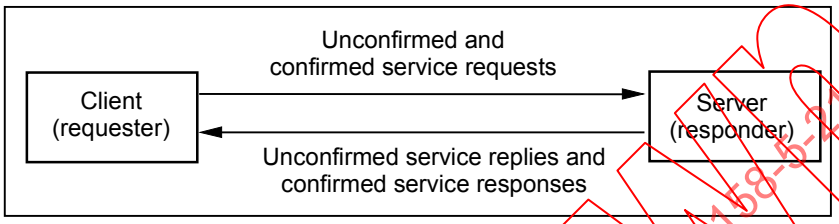


Figure 3 – Client/server interactions

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.

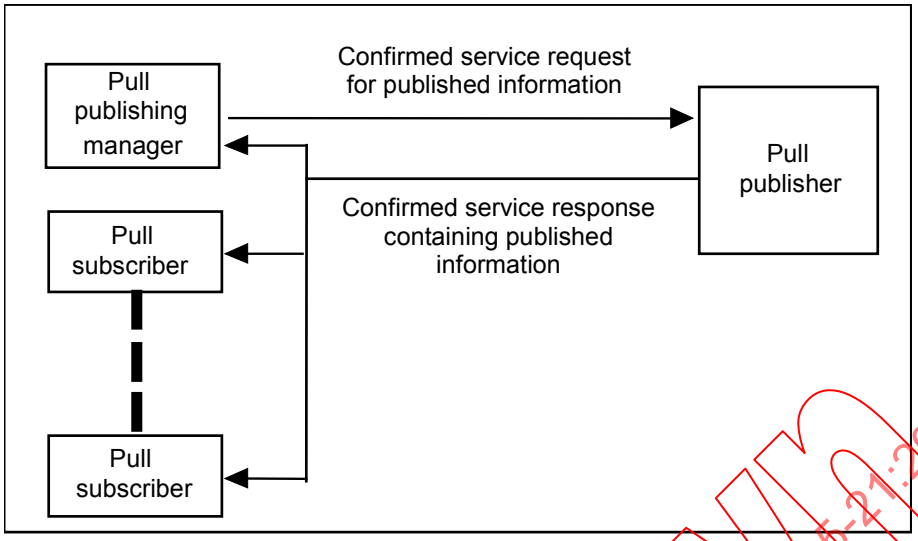


Figure 4 – Pull model interactions

4.1.3.3.3.3.3 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.

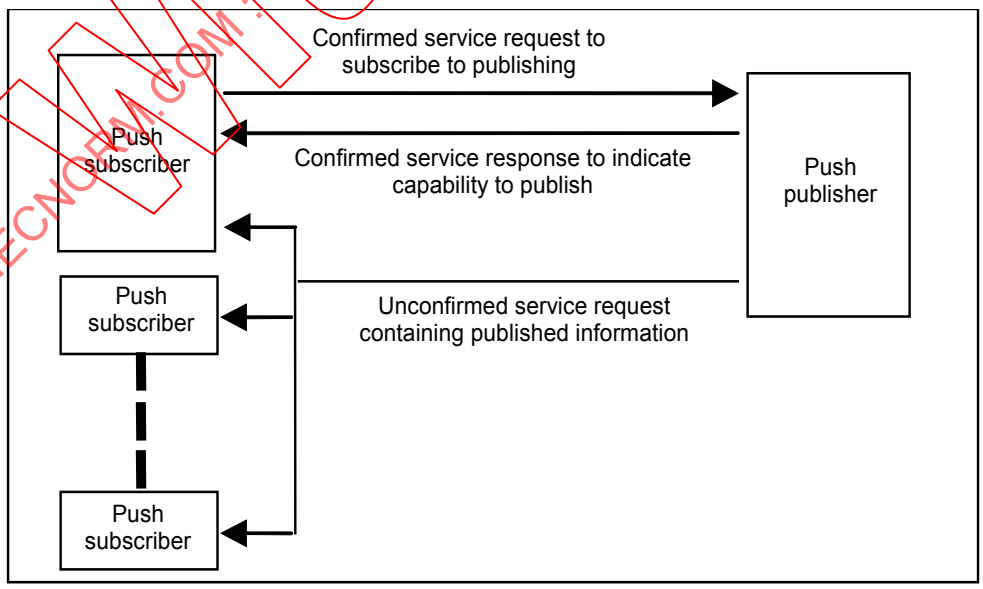


Figure 5 – Push model interactions

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.

4.1.3.4 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.

4.1.3.5 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.

4.1.3.6 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).

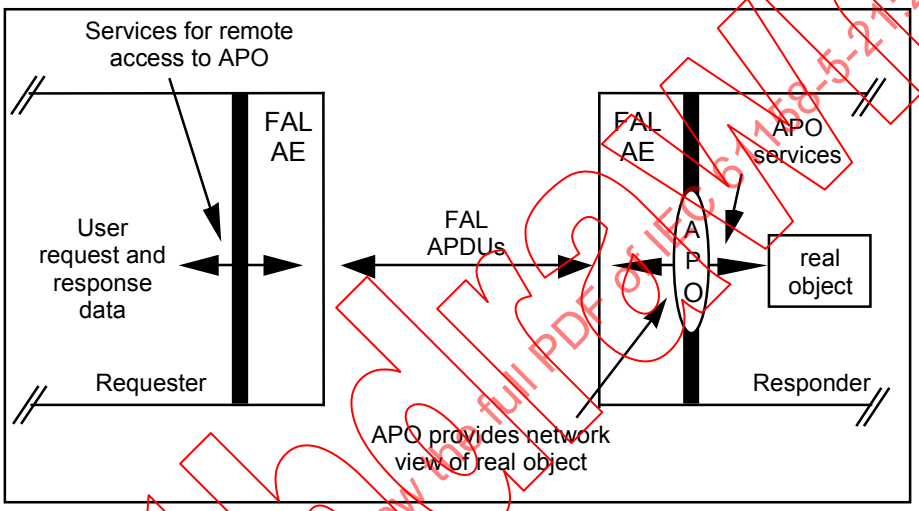


Figure 6 – APOs services conveyed by the FAL

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 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.

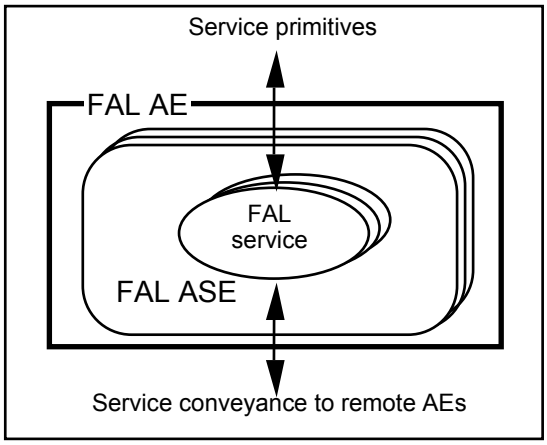


Figure 7 – Application entity structure

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 though 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.

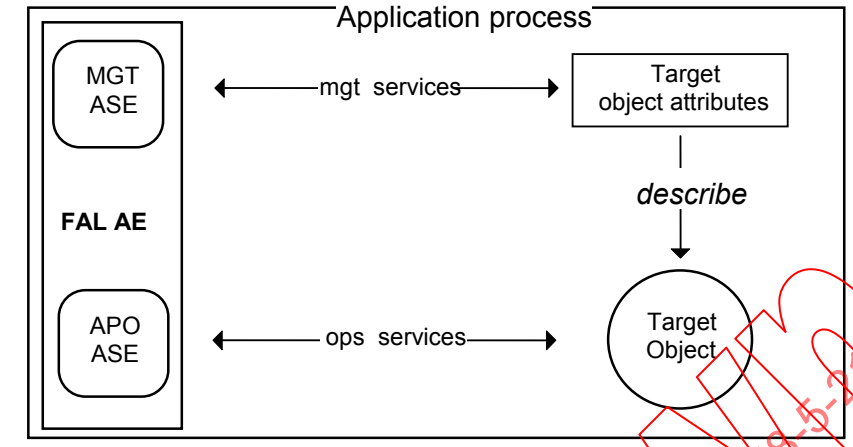


Figure 8 – FAL management of objects

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.

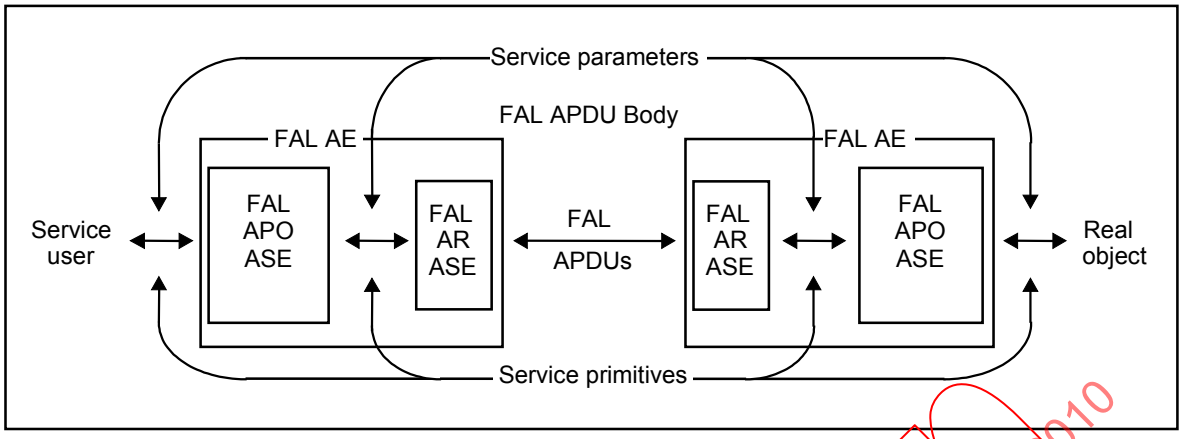


Figure 9 – ASE service conveyance

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.

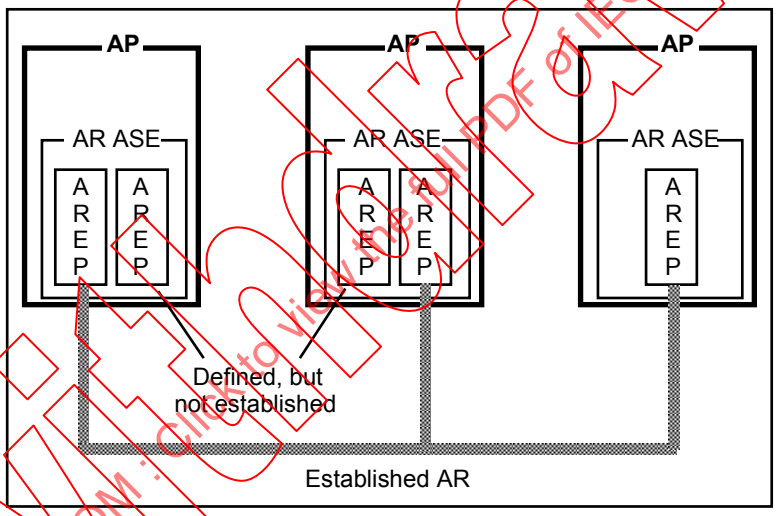


Figure 10 – Defined and established AREPs

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 though 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.

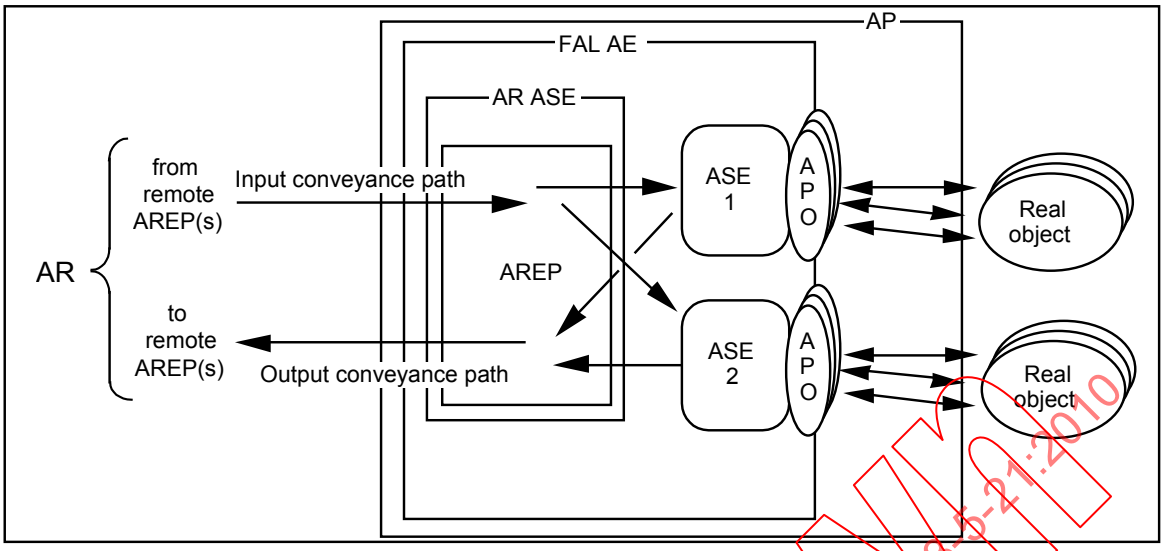


Figure 11 – FAL architectural components

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.

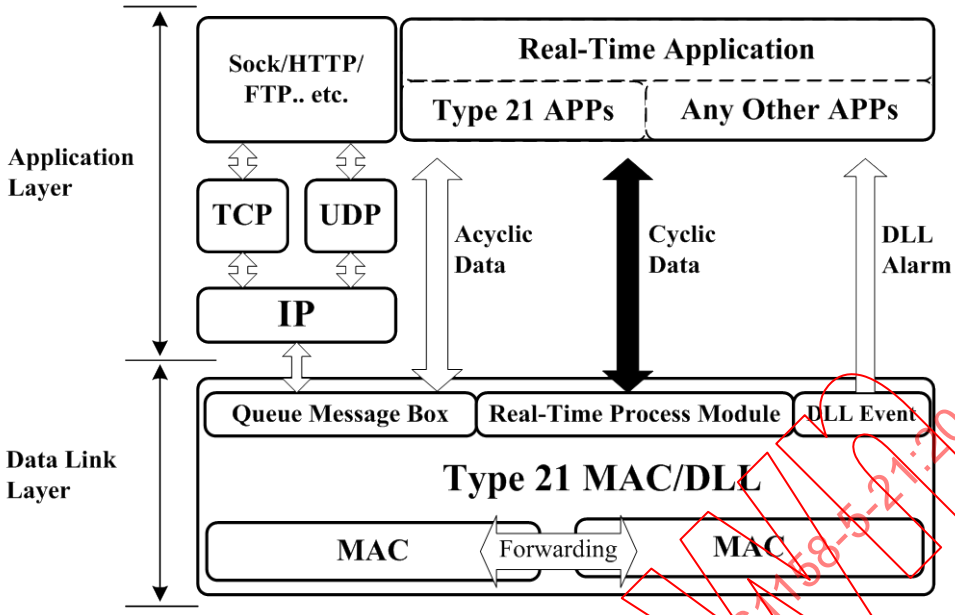


Figure 12 – Interaction between FAL and DLL

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.

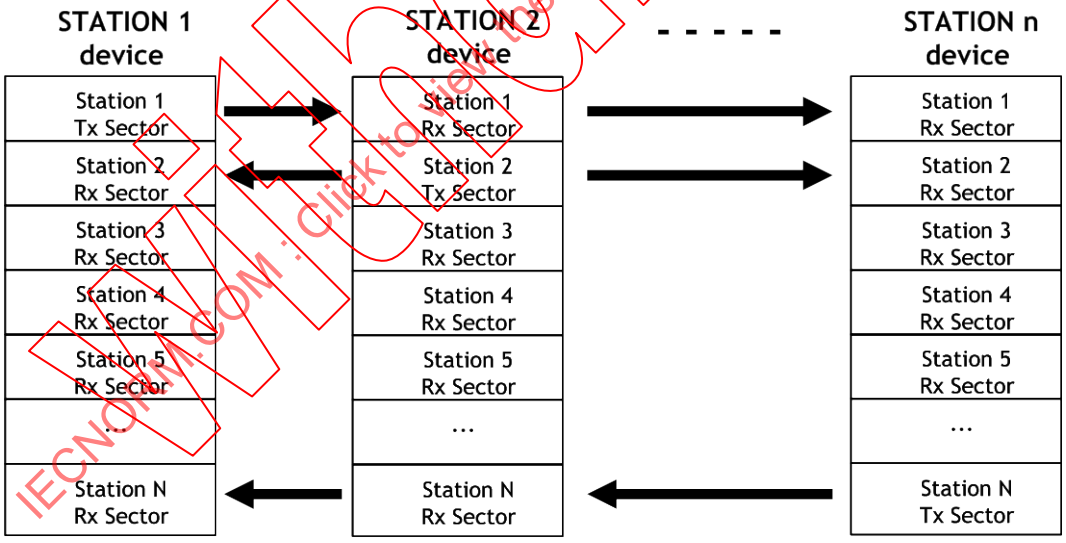


Figure 13 – Publisher-subscriber communication model

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.

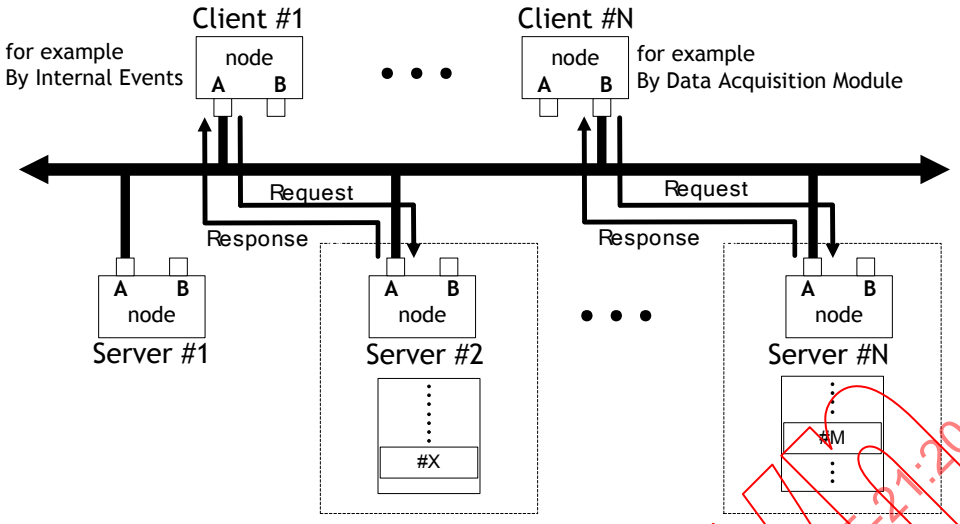


Figure 14 – Client-server communication model

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.

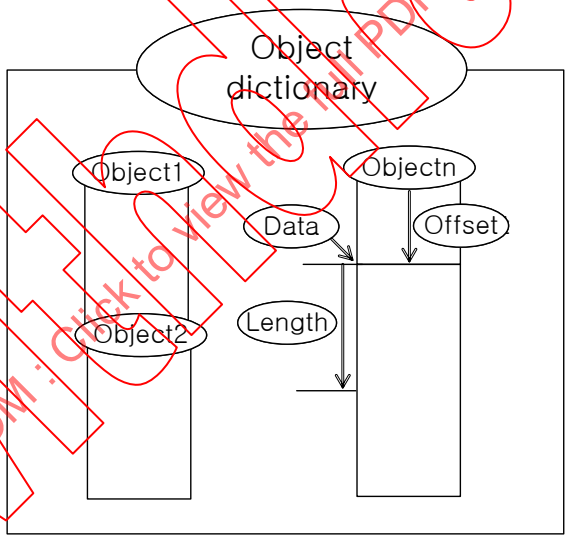


Figure 15 – Object model

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.

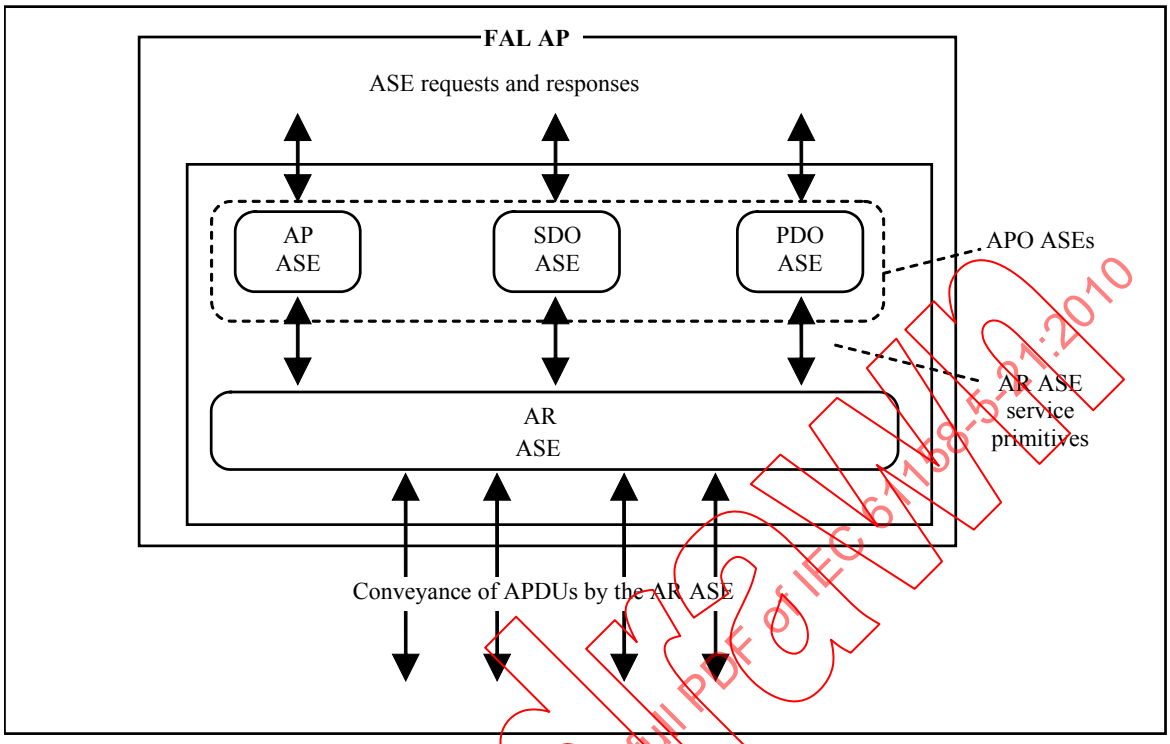


Figure 16 – ASEs of a Type 21 application

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.

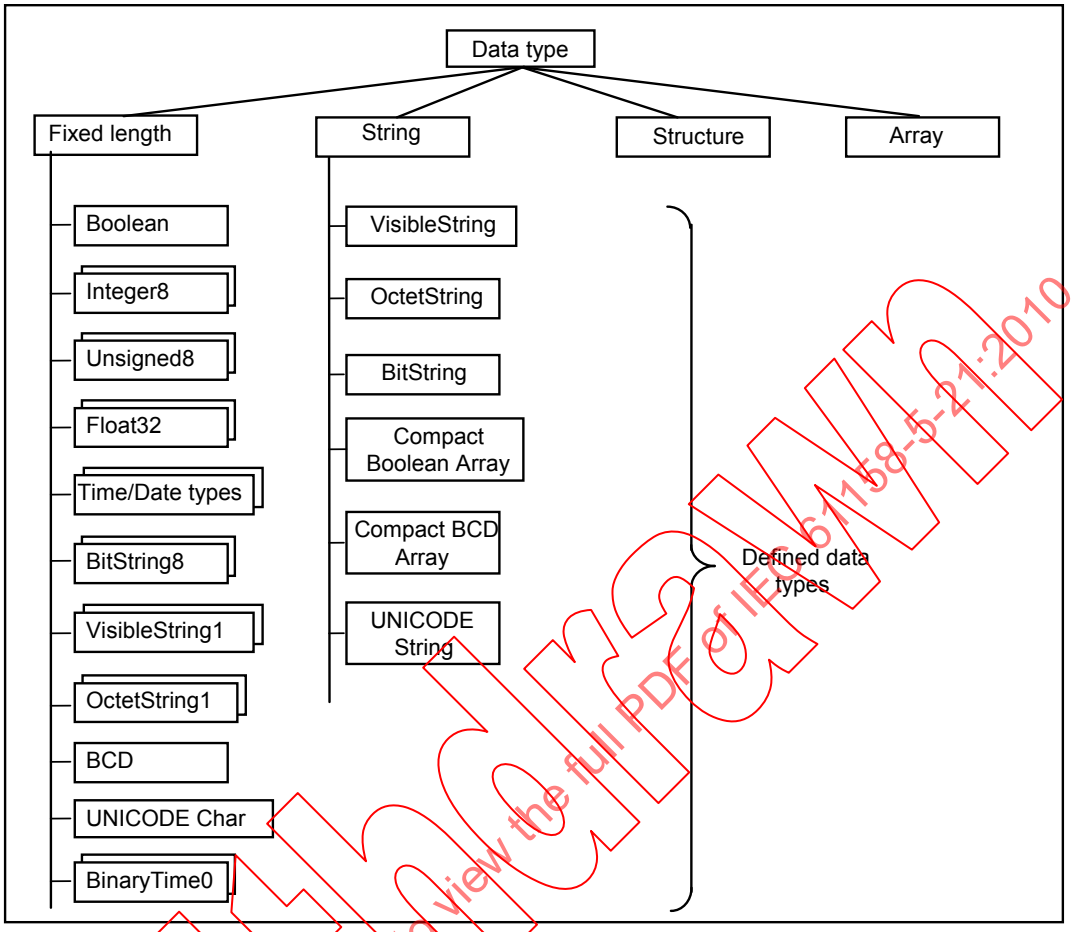


Figure 17 – Data type class hierarchy example

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.

5.3.2.2 VisibleString

CLASS:			Data type
ATTRIBUTES:			
1	Data type numeric identifier	=	9
2	Data type name	=	VisibleString
3	Format	=	STRING
3.1	Octet length	=	1 to n

This type is defined as the ISO/IEC 646 string type.

5.3.2.3 UnicodeString

CLASS:			Data type
ATTRIBUTES:			
1	Data type numeric identifier	=	11
2	Data type name	=	UnicodeString
3	Format	=	STRING
3.1	Octet length	=	1 to n

This type is defined as the UNICODE string type.

5.4 Data type ASE service specification

There are no operational services defined for the type object.

6 Communication model specification

6.1 ASEs

6.1.1 Application process ASE

6.1.1.1 Overview

This standard models a fieldbus AP. fieldbus APs represent the information and processing resources of a system that can be accessed through FAL services.

The ASE in the FAL that provides these services is called an Application Process ASE (AP ASE). In the AP ASE, the AP is modelled and accessed as an APO with a standardized and predefined identifier.

6.1.1.2 AP class specification

6.1.1.2.1 Formal model

The AP class specifies the attributes and services defined for application processes. Its parent class “TOP” indicates the top of the FAL class tree.

ASE: **AP ASE**
CLASS: **AP**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Attribute:	DL–entity identifier
2	(m)	Attribute:	MAC address
3	(m)	Attribute:	Port information
4	(m)	Attribute:	Protocol version
5	(m)	Attribute:	Device type
6	(m)	Attribute:	Device description
7	(m)	Attribute:	Hardware-Version
8	(m)	Attribute:	Serial Number
9	(m)	Attribute:	Software-Version
10	(m)	Attribute:	Software-Date
11	(m)	Attribute:	Vendor ID
12	(m)	Attribute:	Product Code
13	(m)	Attribute:	Device flags
14	(m)	Attribute:	Device state
15	(m)	Attribute:	tx_cnt_normal
16	(m)	Attribute:	tx_cnt_all
17	(m)	Attribute:	rx_cnt_normal
18	(m)	Attribute:	rx_cnt_all
19	(m)	Attribute:	relay_cnt_normal
20	(m)	Attribute:	relay_cnt_all

SERVICES:

1	(m)	OpsService:	Identify
2	(m)	OpsService:	Status

6.1.1.2.2 Attributes

DL–entity identifier
see IEC 61158-4-21:2010, 4.6.5.2

MAC address
see IEC 61158-4-21:2010, 4.6.5.8

Port information
see IEC 61158-4-21:2010, 4.6.5.9

Protocol version
see IEC 61158-4-21:2010, 4.6.5.10

Device type
see IEC 61158-4-21:2010, 4.6.5.11

Device description
see IEC 61158-4-21:2010, 4.6.5.12

Hardware-Version
attribute contains the hardware version of the device

Serial Number
attribute contains the serial number of the device

Software-Version
attribute contains the software version of the device

Software-Date
attribute contains the software date of the device

Vendor ID
attribute contains the vendor ID of the device

Product Code
attribute contains the product code of the device

Device flags
see IEC 61158-4-21:2010, 4.6.5.3

Device state
see IEC 61158-4-21:2010, 4.6.5.4

tx_cnt_normal
attribute contains the value of the transmitted count of normal packets only

tx_cnt_all
attribute contains the value of the transmitted count of both error packets and normal packets

rx_cnt_normal
attribute contains the value of the received count of normal packets

rx_cnt_all
attribute contains the value of the received count of both error packets and normal packets

relay_cnt_normal
attribute contains the value of the relayed count of normal packets

relay_cnt_all
attribute contains the value of the relayed count of both error packets and normal packets

6.1.1.3 AP ASE service specification

6.1.1.3.1 Supported services

This subclause contains the definition of services that are unique to this ASE. The services defined for this ASE are IDENTIFY and STATUS.

6.1.1.3.2 IDENTIFY service

6.1.1.3.2.1 Service overview

This confirmed service may be used to obtain the information from the device.

6.1.1.3.2.2 Service primitives

The service parameters for this service are shown in Table 3.

Table 3 – Identify service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)	—	—
AREP	M	M(=)	—	—
InvokeID	M	M(=)	—	—
	—	—	—	—
Result(+)	—	—	S	S(=)
AREP	—	—	M	M(=)
InvokeID	—	—	M	M(=)
Service status	—	—	M	M(=)
DL–entity identifier	—	—	M	M(=)
MAC address	—	—	M	M(=)
Port information	—	—	M	M(=)
Protocol version	—	—	M	M(=)
Device type	—	—	M	M(=)
Device description	—	—	M	M(=)
Hardware-Version	—	—	M	M(=)
Serial Number	—	—	M	M(=)
Software-Version	—	—	M	M(=)
Software-Date	—	—	M	M(=)
Vendor ID	—	—	M	M(=)
Product Code	—	—	M	M(=)
	—	—	—	—
Result(-)	—	—	S	S(=)
AREP	—	—	M(=)	M(=)
InvokeID	—	—	M(=)	M(=)
Service status	—	—	M	M(=)
Status code	—	—	M	M(=)

Argument
argument conveys the service specific parameters of the service request

AREP
parameter contains information sufficient for local identification of the AREP to 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 simultaneously, the AREP parameter is extended to identify the context as well as the AREP

Service status
parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements

Status code
parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. Available status codes are listed in IEC 61158-6-21:2010.

Result(+)
selection type parameter indicates that the service request succeeded

DL–entity identifier
see IEC 61158-4-21:2010, 4.6.5.2

MAC address
see IEC 61158-4-21:2010, 4.6.5.8

Port information
see IEC 61158-4-21:2010, 4.6.5.9

Protocol version
see IEC 61158-4-21:2010, 4.6.5.10

Device type
see IEC 61158-4-21:2010, 4.6.5.11

Device description
see IEC 61158-4-21:2010, 4.6.5.12

Hardware-Version
parameter contains the hardware version of the device

Serial Number
parameter contains the serial number of the device

Software-Version
parameter contains the software version of the device

Software-Date
parameter contains the software date of the device

Vendor ID
parameter contains the vendor ID of the device.

Product Code
parameter contains the product code of the device.

Result(-)
selection type parameter indicates that the service request failed. The detailed error information is appended in the status code field.

6.1.1.3.2.3 Service procedure

This service procedure is a sequence of two successive confirmed services in opposite directions.

6.1.1.3.3 STATUS service

6.1.1.3.3.1 Service overview

This confirmed service may be used to request the network-visible state from the AP.

6.1.1.3.3.2 Service primitives

The service parameters for this service are shown in Table 4.

Table 4 – Status service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)	—	—
AREP	M	M(=)	—	—
InvokeID	M	M(=)	—	—
	—	—	—	—
Result(+)	—	—	S	S(=)
AREP	—	—	M	M(=)
InvokeID	—	—	M	M(=)
Service status	—	—	M	M(=)
Device flags	—	—	M	M(=)
Device state	—	—	M	M(=)
tx_cnt_normal	—	—	M	M(=)
tx_cnt_all	—	—	M	M(=)
rx_cnt_normal	—	—	M	M(=)
rx_cnt_all	—	—	M	M(=)
relay_cnt_normal	—	—	M	M(=)
relay_cnt_all	—	—	M	M(=)
	—	—	—	—
Result(-)	—	—	S	S(=)
AREP	—	—	M	M(=)
InvokeID	—	—	M	M(=)
Service status	—	—	M	M(=)
Status code	—	—	M	M(=)

Argument
argument conveys the service specific parameters of the service request

AREP
parameter contains information sufficient for local identification of the AREP to 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 simultaneously, the AREP parameter is extended to identify the context as well as the AREP

Service status
parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements

Status code
parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. For some errors, further identification of the error may be provided in the Extended Status parameter below. Available status codes are listed in IEC 61158-6-21:2010.

Result(+)
selection type parameter indicates that the service request succeeded

Device flags
see IEC 61158-4-21:2010, 4.6.5.3

Device state
see IEC 61158-4-21:2010, 4.6.5.4

tx_cnt_normal
parameter contains the value of the transmitted count of normal packets

tx_cnt_all
parameter contains the value of the transmitted count of both error packets and normal packets

rx_cnt_normal
parameter contains the value of the received count of normal packets

rx_cnt_all
parameter contains the value of the received count of both error packets and normal packets

relay_cnt_normal
this parameter contains the value of the relayed count of normal packets

relay_cnt_all
parameter contains the value of the relayed count of both error packets and normal packets

Result(-)
selection type parameter indicates that the service request failed. The detailed error information is appended in the status code field

6.1.1.3.3.3 Service procedure

This service procedure is a sequence of two successive confirmed services (as specified in 4.1.6) in opposite directions.

6.1.2 Service data object ASE

6.1.2.1 Overview

For all transfer types, the client takes the initiative for a transfer. The owner of the accessed object dictionary is the server of the service data object (SDO). Either the client or the server can take the initiative to abort the transfer of a SDO. All commands are confirmed. The remote result parameter indicates the success of the request. In case of a failure, an abort transfer request must be executed.

6.1.2.2 SDO class specification

6.1.2.2.1 Formal model

The AP class specifies the attributes and services defined for application processes. Its parent class “TOP” indicates the top of the FAL class tree.

ASE: **SDO ASE**
CLASS: **SDO**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Object identifier
2	(m)	Attribute:	Size
3	(m)	Attribute:	Data type
4	(m)	Attribute:	Access right

SERVICES:

1	(m)	OpsService:	Read
2	(m)	OpsService:	Write

6.1.2.2.2 Attributes

Object identifier
this attribute contains the numeric identifier of the object to be accessed

Size
this attribute specifies the actual size of the object in octets

Data type
this attribute contains the numeric identifier of the data type

Access right
this attribute contains the type of access defined for the object, as shown in Table 5

Table 5 – Access rights for object

Name	Signification
R	Right to read for the registered password
W	Right to write for the registered password
U	Right to use the registered password

6.1.2.3 SDO ASE service specification

6.1.2.3.1 Supported services

This subclause contains the definition of services that are unique to this ASE. The services defined for this ASE are READ and WRITE.

6.1.2.3.2 READ service

6.1.2.3.2.1 Service overview

This confirmed read service may be used to read the value of an SDO.

6.1.2.3.2.2 Service primitives

The service parameters for this service are shown in Table 6.

Table 6 – Read service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)	—	—
AREP	M	M(=)	—	—
InvokeID	M	M(=)	—	—
Object List Count	M	M(=)	—	—
Object List	M	M(=)	—	—
Object-identifier	M	M(=)	—	—
Data type	M	M(=)	—	—
Offset	M	M(=)	—	—
Length	M	M(=)	—	—
	—	—	—	—
Result(+)	—	—	S	S(=)
AREP	—	—	M	M(=)
Invoke ID	—	—	M	M(=)
Service status	—	—	M	M(=)
OD List Count	—	—	M	M(=)
Object List	—	—	M	M(=)
Object-identifier	—	—	M	M(=)
Data type	—	—	M	M(=)
Offset	—	—	M	M(=)
Length	—	—	M	M(=)
Data	—	—	M	M(=)
	—	—	—	—
Result(-)	—	—	S	S(=)
AREP	—	—	M	M(=)
Invoke ID	—	—	M	M(=)
Service status	—	—	M	M(=)
Status code	—	—	M	M(=)

Argument
argument conveys the service specific parameters of the service request

AREP
parameter contains information sufficient for local identification of the AREP to 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 simultaneously, the AREP parameter is extended to identify the context as well as the AREP

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

Object List Count
parameter indicates the number of objects listed in the service request. Each object is designated with object list arguments described below

Object List
parameter contains a stream of information containing the actual list of objects to be read

Object identifier
parameter identifies the entry of the target object to be read by this service

Data type
parameter identifies the data type

Offset
parameter identifies the local offset address of the data to be read from the object

Length
parameter indicates the number of octets to be read

Service status
parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements

Status code
parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. Available status codes are listed in IEC 61158-6-21:2010.

Result(+)
selection type parameter indicates that the service request succeeded

Data
parameter contains the value of the object that has been read and consists of the number of octets indicated in the length of the response primitive

Result(-)
selection type parameter indicates that the service request failed

6.1.2.3.2.3 Service procedure

This service procedure is a sequence of two successive confirmed services (as specified in 4.1.6) in opposite directions. Write service

6.1.2.3.3 WRITE service

6.1.2.3.3.1 Service overview

This confirmed write service may be used to write the value of an SDO.

6.1.2.3.3.2 Service primitives

The service parameters for this service are shown in Table 7.

Table 7 – Write service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)	—	—
AREP	M	M(=)	—	—
InvokeID	M	M(=)	—	—
Object List Count	M	M(=)	—	—
Object List	M	M(=)	—	—
Object identifier	M	M(=)	—	—
Data type	M	M(=)	—	—
Offset	M	M(=)	—	—
Length	M	M(=)	—	—
Data	M	M(=)	—	—
	—	—	—	—
Result(+)	—	—	S	S(=)
AREP	—	—	M	M(=)
InvokeID	—	—	M	M(=)
Service status	—	—	M	M(=)
	—	—	—	—
Result(-)	—	—	S	S(=)
AREP	—	—	M	M(=)
InvokeID	—	—	M	M(=)
Service status	—	—	M	M(=)
Status code	—	—	M	M(=)

Argument
argument conveys the service specific parameters of the service request

AREP
parameter contains information sufficient for local identification of the AREP to 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 simultaneously, the AREP parameter is extended to identify the context as well as the AREP

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

Object List Count
parameter identifies the number of objects listed in the service request. Each object is designated with the object list argument described below

Object List
parameter contains a stream of information containing the actual list of objects to be written

Object identifier
parameter identifies the entry of the target object to be written by this service

Data type
parameter identifies the data type

Offset
parameter identifies the local offset address of the data to be written to the object

Length
parameter indicates the number of octets to be written

Data
parameter contains the value of the object that has been written, and consists of the number of octets indicated in the length of the request primitive

Service status
parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements

Status code
parameter indicates whether or not the service was processed successfully. If an error occurred, it indicates the type of error. Available status codes are listed in IEC 61158-6-21:2010.

Result(+)
selection type parameter indicates that the service request succeeded

Result(-)
selection type parameter indicates that the service request failed

6.1.2.3.3.3 Service procedure

This service procedure is a sequence of two successive confirmed services (as specified in 4.1.6) in opposite directions.

6.1.3 Process data object ASE

6.1.3.1 Overview

The time-critical data are transferred by the process data objects (PDO) using the publisher-subscriber communication model. The periodically scheduled timer-based (TB) or change-of-state (COS) event-triggered PDO data are multicast to subscribers using the unconfirmed send service.

From the device's point of view, there are two types of PDO usage: data transmission and data reception. The transmission PDOs (TPDOs) and reception PDOs (RPDOs) must be distinguished from each other. Devices supporting TPDOs are called PDO publishers and devices that are able to receive PDOs are called PDO subscribers.

The numbers of supported channels between AREPs may be preconfigured by the application, and the method to configure the relationships between AREPs outside the scope of this standard.

6.1.3.2 PDO class specification

6.1.3.2.1 Formal model

ASE:			**PDO ASE**
CLASS:			**PDO**
CLASS ID:			**not used**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Attribute:	list of RPDO channels
1.1	(m)	Attribute:	channel info
1.2	(m)	Attribute:	list of mapped object entries
1.2.1	(m)	Attribute:	object identifier
1.2.2	(m)	Attribute:	data type
1.2.3	(m)	Attribute:	length

2	(m)	Attribute:	list of TPDO channels
2.1	(m)	Attribute:	channel info
2.2	(m)	Attribute:	list of mapped object entries
2.2.1	(m)	Attribute:	object identifier
2.2.2	(m)	Attribute:	data type
2.2.3	(m)	Attribute:	length

SERVICES:

1	(m)	OpsService:	TB-transfer
2	(m)	OpsService:	COS-transfer

6.1.3.2.2 Attributes

List of RPDO channels
attribute specifies the number of logical channels to receive TB/COS-transfer services. The maximum permissible value for the RPDO channel is 254

Channel info
attribute specifies the AREP relationship between publisher and subscriber

List of mapped OD entries
following attributes specify the OD entries to be mapped to the PDO of the respective channel. A PDO consists of up to 256 mapped object entries

Object identifier
parameter identifies the entry of the target object to be written by this service

Data type
parameter identifies the data type

Length
attribute provides the length of the mapped object in octets

List of TPDO channels
following attributes specify the logical channels for outgoing PDOs

all the following attributes correspond to the equivalents in “list-of-RPDO-channels” and are therefore not repeated here

6.1.3.2.3 Services

TB-transfer
service is used to transfer the TB PDO between devices

COS-transfer
service is used to transfer the COS PDO between devices

6.1.3.3 PDO ASE service specification

6.1.3.3.1 Supported services

This subclause contains the definition of services that are unique to this ASE. The services defined for this ASE are TB-transfer and COS-transfer.

6.1.3.3.2 TB-transfer

6.1.3.3.2.1 Service overview

This service is used to transfer process data objects between devices. At the requesting device, the outgoing TB is composed of the appropriate TB attributes. At the receiving device, the incoming TB is handled according to the TB attributes.

6.1.3.3.2.2 Service primitives

The service parameters for this service are shown in Table 8.

Table 8 – TB-transfer

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
TB PDO	M	M(=)

Argument
argument conveys the service specific parameters of the service request

TB PDO
parameter contains the TB PDO data

6.1.3.3.2.3 Service procedure

This service is performed either as one unconfirmed service or as a sequence of two successive unconfirmed services in opposite directions.

6.1.3.3.3 COS-transfer

6.1.3.3.3.1 Service overview

This service is used to transfer process data objects between devices. At the requesting device, the outgoing COS is composed of the appropriate COS attributes. At the receiving device, the incoming COS is handled according to the COS attributes.

6.1.3.3.3.2 Service primitives

The service parameters for this service are shown in Table 9.

Table 9 – COS-transfer

Parameter	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
COS PDO	M	M(=)

Argument
argument conveys the service specific parameters of the service request

COS PDO
parameter contains the COS data

6.1.3.3.3.3 Service procedure

This service is performed either as one unconfirmed service or as a sequence of two successive unconfirmed services in opposite directions.

6.1.4 Application relationship ASE

6.1.4.1 Overview

6.1.4.1.1 General

In a distributed system, application processes communicate with each other by exchanging application layer messages across well-defined application layer communications channels.

These communication channels are modeled in the FAL as ARs.

ARs are responsible for conveying messages between applications according to specific communications characteristics required by time-critical systems. Different combinations of these characteristics lead to the definition of different types of ARs. The characteristics of ARs are defined formally as attributes of AR endpoint classes.

The messages that are conveyed by ARs are FAL service requests and responses. Each is submitted to the AR ASE for transfer by an FAL ASE that represents the class of the APO being accessed. Figure 18 illustrates this concept.

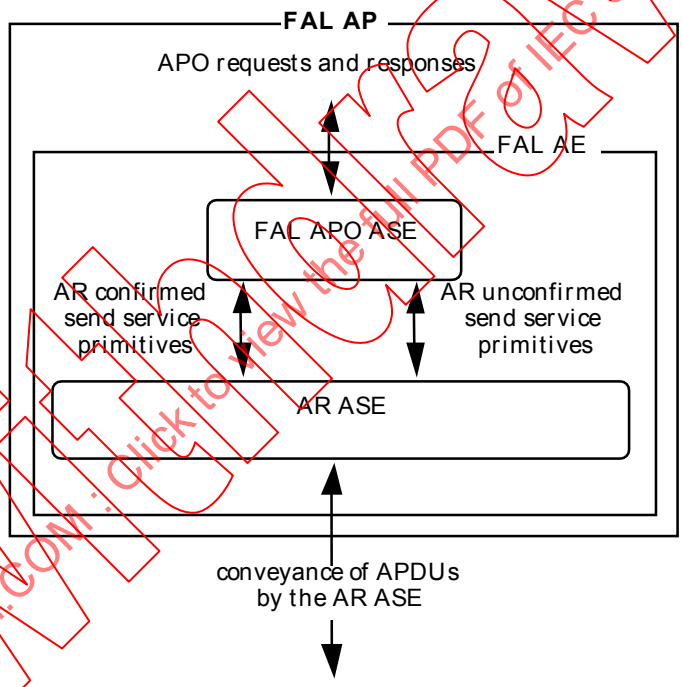


Figure 18 – The AR ASE conveys APDUs between APs

Depending on the type of AR, APDUs may be sent to one or more destination application processes connected by the AR. Other characteristics of the AR determine how APDUs are to be transferred. These characteristics are described below.

6.1.4.1.2 Endpoint context

Each AP involved in an AR contains an endpoint of the AR. Each AR endpoint is defined within the AE of the AP. Its definition defines an AR when combined with the definitions of the other endpoints. To ensure communications compatibility among or between endpoints, each endpoint definition contains a set of compatibility-related characteristics. These characteristics must be configured appropriately for each endpoint for the AR to operate properly.

Endpoint definitions also contain a set of characteristics that describe the operation of the AR. These characteristics define the context of the endpoint when combined with those used to specify compatibility. The endpoint context is used by the AR ASE to manage the operation of the endpoint and the conveyance of APDUs. The characteristics that comprise the endpoint context are described below.

6.1.4.1.2.1 Endpoint role

The role of an AREP determines the permissible behavior of an AP at the AREP. An AREP may have the role of client, server, peer (client and/or server), publisher, or subscriber.

Table 10 and Table 11 summarize the characteristics and combinations of each of the AREP roles.

Table 10 – Conveyance of service primitives by AREP role

	Client	Server	Peer	Publisher	Subscriber
CS Req	X	—	X	—	—
CS Rsp	—	X	X	—	—
UCS Req	—	—	—	X	—

Table 11 – Valid combinations of AREP roles involved in an AR

	Client	Server	Peer	Publisher	Subscriber
Client	—	—	X	—	—
Server	X	—	X	—	—
Peer	X	X	X	—	—
Publisher	—	—	—	—	X
Subscriber	—	—	—	X	—

6.1.4.1.2.2 Cardinality

From the point of view of a client or publisher endpoint, the cardinality of an AR specifies how many remote application processes are involved in an AR. Cardinality is never expressed from the viewpoint of a server or a subscriber.

When expressed from the viewpoint of a client or peer endpoint, ARs are always 1:1. Clients are never capable of issuing a request and waiting for responses from multiple servers.

When expressed from the viewpoint of a publisher endpoint, ARs support multiple subscribers. These ARs are 1:many. Such ARs provide for communications between one application and a group of one or more applications. They are often referred to as multicast.

6.1.4.1.3 Conveyance model

6.1.4.1.3.1 General

The conveyance model defines how APDUs are sent between endpoints of an AR. Three characteristics are used to define these transfers:

a) conveyance paths;

b) trigger policy;

c) conveyance policy.

6.1.4.1.3.2 Conveyance paths

The purpose of AR ASEs is to transfer information between AR endpoints. This information transfer occurs over the conveyance paths of an AR. A conveyance path represents a one-way communication path used by an endpoint for input or output.

The endpoint is configured with either one or two conveyance paths to support the role of the application process. Endpoints that only send or only receive are configured with either a send or receive conveyance path, respectively, and those that do both are configured with both. ARs with a single conveyance path are called unidirectional, while those with two conveyance paths are called bidirectional.

Unidirectional ARs are capable of conveying service requests only. To convey service responses, a bidirectional AR is necessary. Therefore, unidirectional ARs support the transfer of unconfirmed services in one direction only, while bidirectional ARs support the transfer of unconfirmed and confirmed services initiated by only one endpoint, or by both endpoints.

6.1.4.1.3.3 Trigger policy

Trigger policy indicates when APDUs are transmitted by the DLL over the network.

The first type is referred to as user-triggered. User-triggered AREPs submit FAL APDUs to the DLL for transmission at the earliest opportunity.

The second type is referred to as network-scheduled. Network-scheduled AREPs submit FAL APDUs to the DLL for transmission according to a schedule determined by management.

This network scheduling mechanism is cyclic.

6.1.4.1.3.4 Conveyance policy

Conveyance policy indicates whether APDUs are transferred according to a buffer model or a queue model. These models describe the method of conveying APDUs from sender to receiver.

Buffered ARs contain conveyance paths that have a single buffer at each endpoint. Updates to the source buffer are conveyed to the destination buffer according to the trigger policy of the AR. Updates to either buffer overwrite the contents with new data. In buffered ARs, unconveyed or undelivered data are lost once they are overwritten. Data in a buffer may be read multiple times without the contents being destroyed.

Queued ARs contain conveyance paths that are represented as a queue between endpoints. Queued ARs convey data using a FIFO queue. Queued ARs are not overwritten; new entries are queued until they can be conveyed and delivered.

If a queue is full, new messages will not be added.

NOTE The AR conveyance services are described abstractly in such a way that they are capable of being implemented to operate using buffers or queues. These services may be implemented in a number of ways. For example, they may be implemented such that the capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying DLL. Alternatively, these services may be implemented such that these capabilities are combined so that the buffer/queue may be loaded and transferred in a single request. On the receiving side, these services may be implemented by delivering the data when received, or by indicating receipt and allowing the user to retrieve the data in a separate operation. Another option is to require the user to detect that the buffer or queue has been updated.

6.1.4.1.4 Underlying communications services

6.1.4.1.4.1 General

The AR ASE conveys FAL APDUs using the capabilities of the underlying DLL. Several characteristics are used to describe these capabilities. This clause provides a description of each. These characteristics are specific to the data link mapping defined in IEC 61158-6-21:2010. Their precise specification can be found there.

6.1.4.1.4.2 Connection-oriented services

The underlying layer does not support AR endpoints by providing connection-oriented services. Thus, connections have to be preconfigured by means of the application layer.

6.1.4.1.4.3 Buffered and queued services

The underlying layer may support AR endpoints by providing buffered or queued services. These services can be used to implement buffers or queues required by certain classes of endpoint.

6.1.4.1.4.4 Cyclic and acyclic transfers

The underlying layer may support AR endpoints by providing cyclic or acyclic services.

6.1.4.1.5 AR establishment

For an AR endpoint to be used by an application process, the corresponding AR must be active. When an AR is activated, it is referred to as being "established."

ARs can be pre-established. Pre-established means that the AE that maintains the endpoint context is created before the AR is connected to the network. In this case, communications among the applications involved in the AR may take place without first having to establish the AR explicitly.

6.1.4.1.6 Application relationship classes

AREPs are defined with a combination of characteristics to form different classes of ARs.

6.1.4.2 Application relationship class specification

6.1.4.2.1 Formal model

The formal AR endpoint model defines the characteristics common to all AR endpoints. This class is not capable of being instantiated. It is present only for the inheritance of its attributes and services by its subclasses, each specified in a separate clause of this standard. All AR endpoint attributes are accessible through system management.

FAL ASE:			**AR ASE**
CLASS:			**AR ENDPOINT**
CLASS ID:			**not used**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Attribute:	FAL Revision
2	(m)	Attribute:	Dedicated (TRUE, FALSE)
3	(m)	Attribute:	Cardinality (1:1, 1:many)
4	(m)	Attribute:	Conveyance policy (queued, buffered)
5	(m)	Attribute:	Conveyance path (unidirectional, bidirectional)
6	(m)	Attribute:	Trigger policy (user-triggered, network-scheduled)
7	(o)	Attribute:	Transfer Syntax
SERVICES:			
1	(o)	OpsService:	AR-unconfirmed send
2	(o)	OpsService:	AR-Confirmed Send

6.1.4.2.2 Attributes

FAL revision
specifies the revision level of the FAL protocol used by this endpoint. The revision level is in the AR header of all FAL PDUs transmitted

Dedicated
attribute specifies whether the endpoint is dedicated or not. When TRUE, the services of the AR ASE are accessed directly by the FAL-user

Cardinality
attribute specifies the cardinality of the AR described in 4.1.3.7

Conveyance policy
attribute specifies the conveyance policy of the AR described in 4.1.3.7

Conveyance path
attribute specifies the conveyance path of the AR described in 4.1.3.7

Trigger policy
attribute specifies the trigger policy of the AR described in 4.1.3.7

Transfer syntax
optional attribute identifies the encoding rules to be used on the AR. When not present, the default FAL transfer syntax of this standard is used

6.1.4.2.3 Services

All services defined for this class are optional. When an instance of the class is defined, at least one shall be selected.

AR-unconfirmed send
This optional service is used to send an unconfirmed service.

AR-confirmed send
This optional service is used to send a confirmed service.

6.1.4.3 Application relationship ASE service specification

6.1.4.3.1 Supported services

This clause contains the definitions of services that are unique to this ASE. The services defined for this ASE are AR-unconfirmed send and AR-confirmed send.

The AR-confirmed send service contains the FAL PDU body as part of the Result parameter in the response and confirmation primitives. The FAL PDU body may contain either a positive or negative response returned by the FAL-user transparently to the AR ASE. Therefore, these services have a single Result parameter instead of the separate Result(+) and Result(-) parameters commonly used to convey the positive and negative responses returned by the FAL-user.

6.1.4.3.2 AR-unconfirmed send service

6.1.4.3.2.1 Service overview

This service is used to send AR-unconfirmed send request APDUs for FAL APO ASEs. The AR-unconfirmed send service may be requested at either endpoint of a 1:1 bidirectional AR, at the server endpoint of a 1:1 unidirectional AR, or at the publisher endpoint of a 1:many AR.

6.1.4.3.2.2 Service primitives

The service parameters for each primitive are shown in Table 12.

Table 12 – AR-unconfirmed send

Parameter	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Remote DL–entity identifier	M	M(=)
FAL Service Type	M	M(=)
FAL APDU Body	M	M(=)

Remote DL–entity identifier
parameter contains the destination DLSAP address in the request and the source DLSAP address in the indication

FAL service type
parameter contains the type of service being conveyed

FAL APDU body
parameter contains the service-dependent body for the APDU

6.1.4.3.2.3 Service procedure

The AR-unconfirmed send service is a service that operates through a queue.

The requesting FAL ASE submits an AR-unconfirmed send request primitive to its AR ASE. The AR ASE builds an AR-unconfirmed send request APDU.

The AR ASE queues the APDU for submission to the lower layer.

If the AREP is user-triggered, the AR ASE immediately requests the lower layer to transfer the APDU. If the AR is network-scheduled, the AR ASE requests the DLL to transfer the data at the scheduled time. The data link mapping indicates how the AR ASE coordinates its requests to transmit the data with the DLL.

Upon receipt of the AR-unconfirmed send request APDU, the receiving AR ASE delivers an AR-unconfirmed send indication primitive to the appropriate FAL ASE as indicated by the FAL service type parameter.

6.1.4.3.3 AR-confirmed send service

6.1.4.3.3.1 Service overview

This service is used to send confirmed request and response APDUs for FAL APO ASEs. The AR-confirmed send service may be requested at client and peer endpoints of 1:1 bidirectional ARs.

6.1.4.3.3.2 Service primitives

The service parameters for each primitive are shown in Table 12.

Table 13 – AR-confirmed send

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M(=)	—	—
AREP	M	M(=)	—	—
InvokeID	M	M(=)	—	—
Server DL–entity identifier	M	M(=)	—	—
FAL service type	M	M(=)	—	—
FAL APDU body	M	M(=)	—	—
Result	—	—	M	M(=)
AREP	—	—	M	M(=)
InvokeID	—	—	M	M(=)
Client DL–entity identifier	—	—	M	M(=)
FAL service type	—	—	M	M(=)
FAL APDU body	—	—	M	M(=)

Server DL–entity identifier
parameter contains the DL–entity identifier of the server

FAL service type
parameter contains the type of service being conveyed

FAL APDU body
parameter contains the service-dependent body for the APDU

Result
parameter indicates that the service request either succeeded or failed

Client DL–entity identifier
parameter contains the DL–entity identifier of the client

Service procedure

The AR-confirmed send service is a service that operates through a queue.

The requesting FAL ASE submits an AR-confirmed send request primitive to its AR ASE. The AR ASE creates a transaction state machine to control the invocation of the service.

The AR ASE builds an AR-confirmed send request APDU and queues it for submission to the lower layer. Then, the AR ASE immediately requests the lower layer to transfer the APDU.

Upon receipt of the AR-confirmed send request APDU, the receiving AR ASE delivers an AR-confirmed send indication primitive to the appropriate FAL ASE as indicated by the FAL service type parameter.

The responding FAL ASE submits a confirmed send response primitive to its AR ASE. The AR ASE builds a confirmed send response APDU.

The AR ASE queues the APDU for submission to the lower layer. Then, the AR ASE immediately requests the lower layer to transfer the APDU.

Upon receipt of the confirmed send response APDU, the receiving AR ASE uses the InvokeID contained in the response APDU to associate the response with the appropriate request and cancel the associated transaction state machine. The AR ASE delivers an AR-confirmed send confirmation primitive to the requesting FAL ASE.

If the timer expires before the sending AR ASE receives the response APDU, the AR ASE cancels the associated transaction state machine and delivers an AR-confirmed send confirmation(-) primitive to the requesting FAL ASE.

6.2 ARs

6.2.1 Point-to-point user-triggered confirmed client/server AREP (PTC-AR)

6.2.1.1 Class overview

This class is defined to support the on-demand exchange of confirmed services between two or more application processes. It uses connectionless-mode data link services for the exchanges. The behavior of this class is described below.

An AR ASE user submits a request APDU as an AR ASE service data unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

The AREP receiving the request APDU from its underlying layer queues it for delivery to its AR ASE user in the order in which it was received.

For a confirmed service request, the AREP receiving the request APDU accepts the corresponding response APDU from its AR ASE user and queues it to the underlying layer for transfer.

The AREP that issued the request APDU receives the response APDU from its underlying layer and queues it for delivery to its AR ASE user in the order in which it was received.

The following summarizes the characteristics of this AREP class.

Roles	Client
	Server
Cardinality	1:1
Conveyance paths	Bidirectional
Trigger policy	User-triggered
Conveyance policy	Queued

Formal model

FAL ASE:			**AR ASE**
CLASS:			**PTC-AR**
CLASS ID:			**not used**
PARENT CLASS:			**AR ENDPOINT**
ATTRIBUTES:			
1	(m)	Attribute:	Role (CLIENT, SERVER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	Transmit DL Mapping Reference
4	(m)	Attribute:	Receive DL Mapping Reference
SERVICES:			
1	(m)	OpsService:	Confirmed Send

6.2.1.2 Network management attributes

Role
attribute specifies the role of the AREP. Valid values are:

a) client. Endpoints of this type issue confirmed service request-APDUs to servers and receive confirmed service response-APDUs;

b) server. Endpoints of this type receive confirmed service request-APDUs from clients and issue confirmed service response-APDUs to them.

AREP-state
attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-21:2010.

Transmit data link mapping reference
attribute provides a reference to the underlying DLL mapping for the transmit conveyance path for this AREP. Data link mappings for the DLL are specified in IEC 61158-6-21:2010.

Receive data link mapping reference
attribute provides a reference to the underlying DLL mapping for the receive conveyance path for this AREP. Data link mappings for the DLL are specified in IEC 61158-6-21:2010.

6.2.1.3 Services

Confirmed send
optional service is used to send a confirmed service on an AR

6.2.2 Multipoint network-scheduled unconfirmed publisher-subscriber AREP (MSU-AR)

6.2.2.1 Class overview

This class is defined to support the push model for scheduled and buffered distribution of unconfirmed services to one or more application processes.

The behavior of this type of AR can be described as follows.

An AR ASE user submits a request APDU as an AR ASE service data unit to its AREP for distribution. The sending AREP writes the APDU into the internal buffer, overwriting the existing buffer contents.

The AREP transfers the buffer contents at the next scheduled transfer opportunity.

If the AREP receives another APDU before the buffer contents are transmitted, the buffer contents will be overwritten with the new APDU, and the previous APDU will be lost. When the buffer contents are transmitted, the AR ASE notifies the user of transmission.

At the receiving endpoint, the APDU is received from the network and is written immediately into the buffer, overwriting the existing contents of the buffer. The endpoint notifies the user that the APDU has arrived and delivers it to the user according to the local user interface. If the APDU has not been delivered before the next APDU arrives, it will be overwritten by the next APDU and lost.

An FAL-user receiving the buffered transmission may request to receive the currently buffered APDU later.

The following summarizes the characteristics of this AREP class.

Roles	Publisher
	Subscriber
Cardinality	1:n
Conveyance paths	Unidirectional
Trigger policy	Network-scheduled
Conveyance policy	Buffered

Formal model

FAL ASE:	**AR ASE**

CLASS:			**MSU-AR**
CLASS ID:			**not used**
PARENT CLASS:			**AR ENDPOINT**
ATTRIBUTES:			
1	(m)	Attribute:	Role (PUBLISHER, SUBSCRIBER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	DL Mapping Reference
SERVICES:			
1	(m)	OpsService:	Unconfirmed Send

6.2.2.2 Network management attributes

Role
attribute specifies the role of the AREP. Valid values are:

a) push-publisher. Endpoints of this type publish their data issuing unconfirmed service request-APDUs;

b) push-subscriber. Endpoints of this type receive data from service request-APDUs released by a push-publisher AREP.

AREP-state
attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-21:2010.

DL-mapping-reference
for publisher AREPs, this attribute specifies the mapping to the transmit conveyance path. For subscriber AREPs, this attribute specifies the mapping to the receive conveyance path. Data link mapping attributes for the DLL are specified in IEC 61158-6-21:2010.

6.2.2.3 Services

Unconfirmed Send
optional service is used to send an unconfirmed service on an AR

6.2.3 Multipoint user-triggered unconfirmed publisher-subscriber AREP (MTU-AR)

6.2.3.1 Class overview

This class is defined to support the on-demand queued distribution of unconfirmed services to one or more application processes. The behavior of this class is described as follows.

An AR ASE user submits a request APDU as an AR ASE service data unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

The AREP receiving the request APDU from its underlying layer queues it for delivery to its AR ASE user in the order in which it was received.

The following summarizes the characteristics of this AREP class.

Roles	Publisher
	Subscriber
Cardinality	1:n
Conveyance paths	Unidirectional
Trigger policy	User-triggered
Conveyance policy	Queued

Formal model

FAL ASE:	**AR ASE**
CLASS:	**MTU-AR**
CLASS ID:	**not used**

PARENT CLASS:			**AR ENDPOINT**
ATTRIBUTES:			
1	(m)	Attribute:	Role (PUBLISHER, SUBSCRIBER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	DL Mapping Reference
SERVICES:			
1	(m)	OpsService:	Unconfirmed Send

6.2.3.2 Network management attributes

Role
attribute specifies the role of the AREP. Valid values are:

push-publisher: Endpoints of this type publish their data issuing unconfirmed service request-APDUs;

push-subscriber: Endpoints of this type receive data from service request-APDUs released by a push-publisher AREP.

AREP-state
attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-21:2010.

DL-mapping-reference
for publisher AREPs, this attribute specifies the mapping to the transmit conveyance path. For subscriber AREPs, this attribute specifies the mapping to the receive conveyance path. Data link mapping attributes for the DLL are specified in IEC 61158-6-21:2010.

6.2.3.3 Services

Unconfirmed Send
optional service is used to send an unconfirmed service on an AR

6.3 Summary of FAL classes

This clause contains a summary of the defined FAL Classes. The Class ID values have been assigned to be compatible with existing standards. Table 14 provides a summary of the FAL classes.

Table 14 – FAL class summary

FAL ASE	Class
Application Process	AP
Data type	Fixed-length and String
Process data object	PDO
Service data object	SDO
Application Relationship	AREP
—	PTC-AREP
—	MSU-AREP
—	MTU-AREP

6.4 Permitted FAL services by AREP role

Table 15 defines the valid combinations of services and AREP roles. The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 15 – Services by AREP role

	Unc	Cnf	Client		Server		Push Publisher		Push Subscriber	
FAL Services			**req**	**rcv**	**req**	**rcv**	**req**	**rcv**	**req**	**rcv**
AP ASE										
Identify		x	x			x				
Status		x	x			x				
PDO ASE										
TB-transfer	x						x			x
COS-transfer	x						x			x
SDO ASE										
Read		x	x			x				
Write		x	x			x				
AR ASE										
PTC-AR		x	x			x				
MSU-AR	x						x			x
MTU-AR	x						x			x

Bibliography

IEC/TR 61158-1:2010[2], *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

2 To be published.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-21: Définition des services de la couche application – Eléments de Type 21

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de propriété et de ne pas avoir signalé leur existence.

NOTE L'utilisation de certains des types de protocoles associés est limitée par les détenteurs de leurs droits de propriété intellectuelle. Dans tous les cas, l'engagement à un abandon limité des droits de propriété intellectuelle pris par les détenteurs de ces droits permet d'utiliser un type particulier de protocole de couche liaison de données avec des protocoles de couche physique et de couche application dans des combinaisons de types telles que spécifiées de façon explicite dans les parties profil. L'utilisation des divers types de protocoles dans d'autres combinaisons peut exiger la permission donnée par les détenteurs respectifs de leurs droits de propriété intellectuelle.

La Norme internationale CEI 61158-5-21:2010 a été établie par le sous-comité 65C: Réseaux de communication industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

La présente norme annule et remplace IEC/PAS 62573 publiée en 2008. Cette première édition constitue une révision technique.

La présente version bilingue publiée en 2012-01 correspond à la version anglaise monolingue publiée en 2010-08.

Le texte anglais de cette norme est issu des documents 65C/606/FDIS et 65C/620/RVD.

Le rapport de vote 65C/620/RVD donne toute information sur le vote ayant abouti à l'approbation de cette norme.

La version française de cette norme n'a pas été soumise au vote.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Une liste de toutes les parties de la série CEI 61158, présentées sous le titre général *Réseaux de communication industriels – Spécifications des bus de terrain*, est disponible sur le site web de la CEI.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous http://webstore.iec.ch dans les données relatives à la publication recherchée. À cette date, la publication sera:

- reconduite;
- supprimée;
- remplacée par une révisée, ou
- amendée.

NOTE La révision de la présente norme sera synchronisée avec les autres parties de la série CEI 61158.

INTRODUCTION

La présente partie de la CEI 61158 est l'une d'une série produite pour faciliter l'interconnexion de composants d'un système d'automatisation. Elle est apparentée à d'autres normes de la série telle que définie par le modèle de référence des bus de terrain à trois couches décrit dans le rapport IEC/TR 61158-1.

Le service application est fourni par le protocole d'application utilisant les services disponibles de la liaison de données ou autre couche immédiatement inférieure. La présente norme définit les caractéristiques de services d'application pouvant être exploitées par les applications de bus de terrain et/ou la gestion de système.

Dans toute la série de normes relatives aux bus de terrain, le terme "service" se réfère à la capacité abstraite fournie par une couche du Modèle de référence de base de l'Interconnexion des systèmes ouverts (OSI) à la couche immédiatement supérieure. Ainsi, le service de la couche application défini dans la présente norme est un service architectural conceptuel, indépendant des divisions administratives et de mise en œuvre.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-21: Définition des services de la couche application – Eléments de Type 21

1 Domaine d'application

1.1 Vue d'ensemble

La Couche application de bus de terrain (FAL[1]) fournit des programmes d'utilisateur avec un moyen d'accéder à l'environnement de communication du bus de terrain. À cet égard, la FAL peut être considérée comme une fenêtre entre les programmes d'application correspondants.

La présente norme fournit les éléments communs pour les communications de base à temps critique et à temps non critique entre des programmes d'application dans un environnement d'automatisation ainsi que le matériau spécifique au protocole de Type 21. Le terme "à temps critique" sert à représenter la présence d'une fenêtre temporelle, dans les limites de laquelle une ou plusieurs actions spécifiées sont tenues d'être parachevées avec un certain niveau défini de certitude. Le manquement à parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent ces actions, avec le risque concomitant pour l'équipement, la centrale et éventuellement pour la vie humaine.

La présente norme définit, de façon abstraite, le service visible de l'extérieur fourni par la FAL en termes:

a) d'un modèle abstrait pour définir des ressources (objets) d'application capables d'être manipulées par les utilisateurs par l'intermédiaire du service FAL;

b) des actions et événements primitifs du service;

c) des paramètres associés à chaque action primitive et événement primitif, et la forme qu'ils prennent;

d) l'interrelation entre ces actions et événements, et leurs séquences valides.

Le but de la présente norme est de définir les services fournis à

a) l'utilisateur de FAL à la frontière entre l'utilisateur et la couche application du Modèle de référence de bus de terrain;

b) la gestion des systèmes au niveau de la frontière entre la couche liaison application et la gestion des systèmes selon le Modèle de référence de bus de terrain.

La présente norme décrit la structure et les services de la FAL selon la CEI, en conformité avec le Modèle de référence de base de l'OSI (ISO/CEI 7498) et la Structure de la couche application de l'OSI (ISO/CEI 9545).

Les services et protocoles de la FAL sont fournis par des entités d'application (AE[2]) de la FAL contenues dans les processus d'application. L'AE de la FAL se compose d'un jeu d'éléments de service application (ASE[3]) orientés objet et d'une entité de gestion de couche (LME[4]) qui

---

1 FAL = *Fieldbus Application Layer*

2 AE = *Application Entity*

3 ASE = *Application Service Element*

4 LME = *Layer Management Entity*

gère l'AE. Les ASE fournissent des services de communication qui fonctionnent sur un jeu de classes d'objets de processus application (APO[5]) connexes. L'un des ASE de la FAL est un ASE de gestion qui fournit un jeu commun de services pour la gestion des instances de classes de la FAL.

Bien que ces services spécifient la manière dont les demandes et les réponses sont émises et délivrées, ils n'incluent pas une spécification de ce que les applications qui demandent et qui répondent doivent en faire. Autrement dit, ces services définissent quelles applications de demandes et de réponses peuvent envoyer ou recevoir, mais pas les fonctions des applications elles-mêmes. Cela permet une plus grande flexibilité aux utilisateurs de la FAL pour normaliser un tel comportement d'objet. En plus de ces services, certains services d'appui sont également définis dans la présente norme pour fournir l'accès à la FAL afin de maîtriser certains aspects de son fonctionnement.

1.2 Spécifications

L'objectif principal de la présente norme est de spécifier les caractéristiques des services conceptuels d'une couche application qui sont adaptées à des communications à temps critique et, donc, complètent le Modèle de référence de base de l'OSI en guidant le développement des protocoles de couche application pour les communications à temps critique.

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles industriels de communication préexistants. Ce dernier objectif donne naissance à la diversité des services normalisés comme les divers types de la CEI 61158, et les protocoles correspondants normalisés dans les sous-parties de la CEI 61158-6.

La présente norme peut être utilisée comme la base pour les interfaces formelles de programmation d'applications. Néanmoins, elle n'est pas une interface de programmation formelle et il faut pour toute interface de ce type traiter de questions de mise en œuvre qui ne sont pas couvertes par la présente norme, y compris:

a) les tailles et l'ordonnancement des octets pour les divers paramètres de service à plusieurs octets;

b) la corrélation de primitives appariées pour la demande et la confirmation ou pour l'indication et la réponse.

1.3 Conformité

La présente norme ne spécifie aucune mise en œuvre ou aucun produit individuels, de même qu'elle ne restreint nullement les mises en œuvre des entités de couche application dans les systèmes d'automation industriels.

Il n'y a pas de conformité d'équipement à la présente norme de définition des services de couche application. Au contraire, la conformité est obtenue par une mise en œuvre de protocoles conformes de couche application qui satisfont à tout type donné de services de couche application définis dans la présente norme.

2 Références normatives

Les documents de référence suivants sont indispensables pour l'application du présent document. Pour les références datées, seule l' citée s'applique. Pour les références non datées, la dernière du document de référence s'applique (y compris les éventuels amendements).

---

5 APO = *Application Process Object*

CEI 60559, *Arithmétique binaire en virgule flottante pour systèmes à microprocesseurs*

CEI 61158-2:2010[6], *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition* (disponible uniquement en anglais)[7]

CEI 61158-3-21:2010[6], *Industrial communication networks – Fieldbus specifications – Part 3-21: Data-link layer service definition – Type 21 elements* (disponible uniquement en anglais)

CEI 61158-4-21:2010[6], *Industrial communication networks – Fieldbus specifications – Part 4-21: Data-link layer protocol specification – Type 21 elements* (disponible uniquement en anglais)

CEI 61158-6-21:2010[6], *Industrial communication networks – Fieldbus specifications – Part 6-21: Application layer protocol specification – Type 21 elements* (disponible uniquement en anglais)

ISO/CEI 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Le modèle de base*

ISO/CEI 7498-3, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Dénomination et adressage*

ISO/CEI 8822, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Définition du service de présentation*

ISO/CEI 9545, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Structure de la couche application*

ISO/CEI 10731:1994, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base – Conventions pour la définition des services OSI*

3 Termes, définitions, symboles, abréviations, et conventions

3.1 Termes et définitions issus d'autres normes ISO/CEI

3.1.1 Termes de l'ISO/CEI 7498-1

a) entité d'application

b) processus d'application

c) unité de données de protocole application

d) élément de service application

e) invocation d'entité d'application

f) invocation de processus d'application

g) transaction d'application

h) système ouvert réel

i) syntaxe de transfert

3.1.2 Termes de l'ISO/CEI 8822

a) syntaxe abstraite

---

6 À paraître.

7 Les publications monolingues des séries IEC 61158 et IEC 61784 sont actuellement en cours de traduction.

b) contexte de présentation

3.1.3 Termes de l'ISO/CEI 9545

a) association d'applications (*application-association*)
b) contexte d'application *(application-context*)
c) nom de contexte d'application *(application context name*)
d) invocation d'entité d'application *(application-entity-invocation*)
e) type d'entité d'application *(application-entity-type*)
f) invocation de processus d'application *(application-process-invocation*)
g) type de processus d'application *(application-process-type*)
h) élément de service application *(application-service-element*)
i) élément de service de contrôle d'application *(application control service element*)

3.2 Termes de la couche liaison de données de bus de terrain

Pour les besoins du présent document, les termes suivants tels que définis dans la CEI 61158-3-21:2010 et dans la CEI 61158-4-21:2010 s'appliquent.

a) DL-Time (heure de liaison de données)
b) DL-Scheduling-policy (politique de programmation de liaison de données)
c) DLCEP
d) DLC
e) DL-connection-oriented mode (mode orienté connexion de liaison de données)
f) DLPDU
g) DLSDU
h) DLSAP
k) link (liaison)
l) ISO/IEC 8802-3:2000 MAC address (adresse MAC selon l'ISO/CEI 8802-3:2000)
m) DL–entity identifier (identificateur d'entité de liaison de données)

3.3 Définitions spécifiques à la couche application des bus de terrain

3.3.1
application
fonction ou structure de données pour laquelle les données sont consommées ou produites

3.3.2
objets d'application
classes d'objets multiples qui gèrent et assurent l'échange de messages pendant le mode exécution à travers le réseau et à l'intérieur du dispositif de réseau

3.3.3
processus d'application
partie d'une application répartie sur un réseau, qui est située sur un dispositif et adressée sans ambiguïté

3.3.4
identificateur de processus d'application
distingue de multiples processus d'application utilisés dans un dispositif

3.3.5
objet de processus d'application
composant d'un processus d'application qui est identifiable et accessible par l'intermédiaire d'une relation d'applications de la FAL

NOTE Les définitions d'objet de processus d'application se composent d'un jeu de valeurs pour les attributs de leur classe (voir la définition de "classe d'objets de processus d'application"). Les définitions d'objet de processus d'application sont accessibles à distance à l'aide des services de l'ASE Gestion d'objet de la FAL. Les services de Gestion d'objet de la FAL peuvent être utilisés pour charger ou mettre à jour des définitions d'objet, pour lire des définitions d'objet, et pour créer et supprimer de manière dynamique des objets d'application et leurs définitions correspondantes.

3.3.6
classe d'objets de processus d'application
classe d'objets de processus d'application définie en termes du jeu de leurs attributs et services accessibles par le réseau

3.3.7
relation d'applications
association coopérative entre deux ou plusieurs invocations d'entités d'application (application-entity-invocation) à des fins d'échange d'informations et de coordination de leur fonctionnement conjoint

NOTE Cette relation est activée soit par l'échange d'unités de données de protocole d'application, soit suite à des activités de préconfiguration.

3.3.8
élément de service d'application de relation d'applications
élément de service d'application qui fournit le moyen exclusif d'établir et de faire cesser toutes les relations d'applications

3.3.9
point d'extrémité de relation d'applications
contexte et comportement d'une relation d'applications tels que vus et maintenus par l'un des processus d'application impliqués dans la relation d'applications

NOTE Chaque processus d'application impliqué dans la relation d'applications maintient son propre point d'extrémité de relation d'applications.

3.3.10
attribut
description d'une caractéristique visible de l'extérieur d'un objet

NOTE Les attributs d'un objet contiennent de l'information relative à des parties variables d'un objet. Typiquement, ils fournissent des informations de statut ou régissent le fonctionnement d'un objet. Des attributs peuvent aussi avoir une incidence sur le comportement d'un objet. Les attributs se répartissent en attributs de classes et attributs d'instances.

3.3.11
comportement
indication de la façon dont un objet réagit à des événements particuliers

3.3.12
voie
simple liaison physique ou logique d'un objet d'application d'entrée ou de sortie d'un serveur au processus

3.3.13
classe
ensemble d'objets, qui représentent tous le même type de composant système

NOTE Une classe est une généralisation d'un objet, un modèle pour définir des variables et des méthodes. Tous les objets dans une classe ont une forme et un comportement identiques, mais contiennent en général des données différentes dans leurs attributs

3.3.14
attribut de classe
attribut partagé par tous les objets au sein de la même classe

3.3.15
code de classe
identificateur unique attribué à chaque classe d'objets

3.3.16
service spécifique à une classe
service défini par une classe d'objets particulière pour accomplir une fonction requise qui n'est pas accomplie par un service commun

NOTE Un objet spécifique à une classe est unique pour la classe d'objets qui le définit.

3.3.17
client
a) objet qui utilise les services d'un autre objet (serveur) pour accomplir une tâche

b) initiateur d'un message auquel un serveur réagit

3.3.18
consommer
acte consistant à recevoir des données d'un producteur

3.3.19
consommateur
nœud ou puits qui reçoit des données d'un producteur

3.3.20
application consommatrice
application qui consomme des données

3.3.21
trajet d'acheminement
flux unidirectionnel d'unités APDU à travers une relation d'applications

3.3.22
cyclique
répétitif d'une manière régulière

3.3.23
cohérence de données
moyen pour une émission et un accès cohérents de l'objet de donnée d'entrée ou de sortie entre client et serveur et au sein du client et du serveur

3.3.24
dispositif
équipement matériel physique relié à la liaison

NOTE Un dispositif peut contenir plus d'un nœud.

3.3.25
profil de dispositif
ensemble d'informations et de fonctionnalité dépendant du dispositif qui assure la cohérence entre des dispositifs similaires relevant du même type de dispositif

3.3.26
informations de diagnostic
toutes les données disponibles au niveau du serveur à des fins de maintenance

3.3.27
nœud d'extrémité
nœud producteur ou consommateur

3.3.28
point d'extrémité
l'une des entités en communication impliquées dans une connexion

3.3.29
erreur
discordance entre une valeur ou un état calculé(e), observé(e) ou mesuré(e) et la valeur ou l'état spécifié(e) ou théoriquement correct(e)

3.3.30
classe d'erreurs
regroupement général pour des définitions d'erreurs connexes et des codes d'erreurs correspondants

3.3.31
code d'erreur
identification d'un type spécifique d'erreur dans une classe d'erreurs

3.3.32
événement
instance d'un changement de conditions

3.3.33
variable FIFO
classe d'objets variables composée d'un jeu d'éléments de type homogène, dans laquelle le premier élément écrit est le premier élément qui peut être lu

NOTE Dans un système de bus de terrain, un seul élément commun peut être transféré suite à une invocation de service.

3.3.34
trame
synonyme simplifié pour l'unité de données de protocole de liaison de données (DLPDU, *data link protocol data unit*)

3.3.35
groupe

a) (Généralités): terme général pour un ensemble d'objets

b) (Adressage): lors de la description d'une adresse, adresse qui identifie plus d'une entité

3.3.36
invocation
acte consistant à utiliser un service ou une autre ressource d'un processus d'application

NOTE Chaque invocation représente un processus distinct de commande qui peut être décrit par son contexte. Une fois que le service s'achève ou que l'utilisation de la ressource est libérée, l'invocation cesse d'exister. Pour des invocations de service, un service a été lancé, mais n'est pas encore achevé s'appelle invocation de service en cours. Pour des invocations de service, un "Invoke ID" peut être utilisé pour identifier sans ambiguïté l'invocation de service et la différencier d'autres invocations de service en cours.

3.3.37
index
adresse d'un objet au sein d'un processus d'application

3.3.38
instance
occurrence physique réelle d'un objet au sein d'une classe qui identifie l'un des plusieurs objets dans la même classe d'objets

EXEMPLE "California" (La Californie) est une instance de la classe d'objets "US-state" (état américain).

NOTE Les termes "objet", "instance" et "instance d'objet" sont utilisés pour se référer à une instance spécifique.

3.3.39
attribut d'instance
attribut qui est propre à une instance d'objet et n'est pas partagé par la classe d'objets

3.3.40
instancié
objet qui a été créé dans un dispositif

3.3.41
dispositif logique
classe spécifique de la FAL qui abstrait un composant logiciel ou un composant de firmware comme une fonction monobloc autonome d'un dispositif d'automation

3.3.42
ID de fabricant
identification de chaque fabrication de produit par un numéro unique

3.3.43
informations de gestion
informations accessibles par le réseau qui appuient la gestion du fonctionnement du système de bus de terrain, y compris la couche application

NOTE La gestion inclut les fonctions telles que la commande, la surveillance et le diagnostic.

3.3.44
réseau
ensemble de nœuds reliés par un certain type de support de communication, notamment des répétiteurs intermédiaires, ponts, routeurs et passerelles de couche inférieure

3.3.45
objet
représentation abstraite d'un composant particulier au sein d'un dispositif, habituellement un ensemble de données connexes sous la forme de variables, et de méthodes (procédures) pour effectuer des opérations sur les données en question qui ont une interface et un comportement clairement définis

3.3.46
dictionnaire d'objets
ensemble de définitions, d'attributs et de paramètres spécifiques aux communications et de données dépendant de l'application

3.3.47
service spécifique à un objet
service propre à la classe d'objets qui le définit

3.3.48
dispositif physique
dispositif d'automation ou autre dispositif de réseau

3.3.49
connexion point à point
connexion qui existe entre exactement deux objets d'application

3.3.50
point d'extrémité d'AR préétabli
point d'extrémité d'AR qui est mis dans un état établi pendant la configuration des AE qui commandent ses points d'extrémité

3.3.51
donnée(s) de processus
objet(s) déjà prétraité(s) et transféré(s) de façon cyclique à des fins d'informations ou de traitement ultérieur

3.3.52
produire
acte consistant à envoyer des données devant être reçues par un consommateur

3.3.53
producteur
nœud qui est chargé d'envoyer des données

3.3.54
propriété
terme général pour les informations descriptives relatives à un objet

3.3.55
fournisseur
source d'une connexion de données

3.3.56
publicateur
rôle d'un point d'extrémité d'AR qui émet des APDU sur le bus de terrain en vue de leur consommation par un ou plusieurs abonnés

NOTE Un publicateur peut ne pas connaître l'identité des abonnés ou leur nombre.

3.3.57
gestionnaire de publication
rôle d'un point d'extrémité d'AR dans lequel il émet une ou plusieurs unités de donnée de protocole application (APDU, *application protocol data unit*) de demande de services confirmés vers un publicateur pour demander qu'un objet spécifié soit édité. La présente norme définit deux types de gestionnaires de publication, à savoir les gestionnaires de publication en en mode pull et les gestionnaires de publication en mode push, chacun de ceux-ci étant défini séparément

3.3.58
publicateur en mode push
type de publicateur qui édite un objet dans une APDU de demande de services non confirmés

3.3.59
gestionnaire de publication en mode push
type de gestionnaire de publication qui demande qu'un objet spécifié soit publié en utilisant un service non confirmé

3.3.60
abonné en mode push
type d'abonné qui reconnaît comme étant des données d'objet éditées les APDU de demande de services non confirmés qu'il a reçues

3.3.61
serveur
a) rôle d'un point d'extrémité de relation d'applications (AREP, *application relationship endpoint* (AREP) dans lequel il retourne au client qui a lancé la demande une APDU confirmée de réponse de service

b) objet qui fournit des services à un autre objet (client)

3.3.62
service
opération ou fonction accomplie par un objet et/ou une classe d'objets à la demande d'un autre objet et/ou d'une autre classe d'objets

3.3.63
poste
hôte d'un AP, identifié par une adresse unique de point d'extrémité de connexion de liaison de données (DLCEP, *data link connection endpoint*)

3.3.64
abonné
rôle d'un AREP dans lequel il reçoit des unités APDU produites par un publicateur

3.4 Abréviations et symboles

AE	Application entity (Entité d'application)
AL	Application Layer (Couche application)
ALME	Application Layer Management Entity (Entité de gestion de couche application)
ALP	Application Layer Protocol (Protocole de couche application)
APO	Application Object (Objet d'application)
AP	Application Process (Processus d'application)
APDU	Application Protocol Data Unit (Unité de données de protocole d'application)
AR	Application Relationship (Relation d'applications)
AREP	Application Relationship End Point (Point d'extrémité de relation d'applications)
ASCII	American Standard Code for Information Interchange (Code américain normalisé pour l'échange d'information)
ASE	Application Service Element (Élément de service d'application)
Cnf	Confirmation
DL-	(comme préfixe) de liaison de données
DLCEP	Data Link Connection End Point (Point d'extrémité de connexion de liaison de données)
DLL	Data Link Layer (Couche liaison de données)
DLM	Data Link Management (Gestion de liaison de données)
DLSAP	Data link Service Access Point (Point d'accès au service liaison de données)
DLSDU	DL-service-data-unit (Unité de données de service liaison de données)
DNS	Domain Name Service (Service de noms de domaine)

FAL	Fieldbus application layer (Couche application de bus de terrain)
Ind	Indication
Req	Request (Demande)
Rsp	Response (Réponse)

3.5 Conventions

3.5.1 Vue d'ensemble

La couche FAL est définie comme un jeu d'éléments ASE orientés objet. Chaque ASE est spécifié dans un paragraphe distinct. Chaque spécification d'ASE est constituée de deux parties: sa spécification de classe et sa spécification de service.

La spécification de classe définit les attributs de la classe. L'accès à ces attributs ne relève pas du domaine d'application du présent document, sauf spécification contraire. La spécification de services définit les services fournis par l'ASE.

3.5.2 Conventions générales

La présente norme utilise les conventions descriptives données dans l'ISO/CEI 10731.

3.5.3 Conventions pour les définitions de classes

Les définitions de classes sont décrites à l'aide de modèles. Chaque modèle est constitué d'une liste d'attributs de la classe. La forme générale du modèle est telle que montrée ci-dessous:

FAL ASE:			**Nom de l'ASE**
CLASS:			**Nom de la classe**
CLASS ID:			**#**
PARENT CLASS:			**Nom de la classe parent**
ATTRIBUTS:			
1	(o)	Attribut clé:	identificateur numérique
2	(o)	Attribut clé:	nom
3	(m)	Attribut:	nom d'attribut(valeurs)
4	(m)	Attribut:	nom d'attribut(valeurs)
4.1	(s)	Attribut:	nom d'attribut(valeurs)
4.2	(s)	Attribut:	nom d'attribut(valeurs)
4.3	(s)	Attribut:	nom d'attribut(valeurs)
5	(c)	Contrainte:	expression de la contrainte
5.1	(m)	Attribut:	nom d'attribut(valeurs)
5.2	(o)	Attribut:	nom d'attribut(valeurs)
6	(m)	Attribut:	nom d'attribut(valeurs)
6.1	(s)	Attribut:	nom d'attribut(valeurs)
6.2	(s)	Attribut:	nom d'attribut(valeurs)
SERVICES:			
1	(o)	OpsService:	nom du service
2	(c)	Contrainte:	expression de la contrainte
2.1	(o)	OpsService:	nom du service
3	(m)	MgtService:	nom du service

(1) FAL ASE: l'entrée est le nom de l'ASE de la FAL qui fournit les services pour la classe spécifiée.

(2) CLASS: l'entrée est le nom de la classe spécifiée. Tous les objets définis à l'aide de ce modèle seront une instance de cette classe. La classe peut être spécifiée par la présente norme ou par un utilisateur de la présente norme.

(3) CLASS ID: l'entrée est un nombre qui identifie la classe spécifiée. Ce nombre n'est pas utilisé pour les éléments de Type 21.

(4) PARENT CLASS: l'entrée est le nom de la classe parent pour la classe spécifiée. Tous les attributs définis pour la classe parent et hérités par celle-ci sont hérités pour la classe définie, et ils n'ont donc pas à être redéfinis dans le modèle pour cette classe.

NOTE La classe parent "TOP" indique que la classe définie est une définition de classe initiale. La classe parent "TOP" est utilisée comme point de départ à partir duquel toutes les autres classes sont définies. L'usage de "TOP" est réservé pour les classes définies par la présente norme.

(5) L'étiquette "ATTRIBUTES" (Attributs) indique que les entrées suivantes sont des attributs définis pour la classe.

a) Chacune des entrées d'attribut contient un numéro de ligne dans la colonne 1; un indicateur obligatoire (m), facultatif (o), conditionnel (c) ou sélecteur (s) dans la colonne 2; une étiquette de type d'attribut dans la colonne 3; un nom ou une expression conditionnelle dans la colonne 4; et une liste facultative de valeurs énumérées dans la colonne 5. Dans la colonne suivant la liste de valeurs, la valeur par défaut pour l'attribut peut être spécifiée.

b) Les objets sont normalement identifiés par un identificateur numérique et/ou par un nom d'objet. Dans les modèles de classe, ces attributs clés sont définis sous l'attribut clé.

c) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par période. Les numéros en dessous se réfèrent à la forme générale de modèle ci-dessus. L'imbrication est utilisée pour spécifier:

i) des champs d'un attribut structuré (4.1, 4.2, 4.3);

ii) des attributs conditionnés à un énoncé de contrainte. Les attributs peuvent être obligatoires (5.1) ou facultatifs (5.2) si la contrainte est vraie. Tous les attributs facultatifs n'exigent pas des énoncés de contraintes comme le fait l'attribut défini en (5.2);

iii) les champs sélection d'un attribut de type choix (6.1 et 6.2).

(6) L'étiquette "SERVICES" indique que les entrées suivantes sont des services définis pour la classe.

a) Un (m) dans la colonne 2 indique que le service est obligatoire pour la classe, alors qu'un (o) indique qu'il est facultatif. Un (c) dans cette colonne indique que le service est conditionnel. Lorsque tous les services définis pour une classe le sont comme étant facultatifs, l'un au moins doit être sélectionné quand une instance de la classe est définie.

b) L'étiquette "OpsService" désigne un service opérationnel (1).

c) L'étiquette "MgtService" désigne un service de gestion (2).

d) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par un point. L'imbrication dans la liste de services sert à spécifier des services conditionnés à un énoncé de contrainte.

3.5.4 Conventions pour les définitions des services

3.5.4.1 Généralités

Le modèle de service, les primitives de service et les diagrammes de séquence temporelle utilisés sont des descriptions totalement abstraites; ils ne constituent pas une spécification pour une mise en œuvre.

3.5.4.2 Paramètres du service

Les primitives de service sont utilisées pour représenter les interactions entre utilisateur de service et fournisseur de service (ISO/CEI 10731). Elles acheminent des paramètres qui indiquent des informations disponibles dans l'interaction entre utilisateur et fournisseur. Dans

n'importe quelle interface particulière, ce ne sont pas tous les paramètres qui doivent être énoncés de façon explicite.

La définition de service selon la présente norme utilise un format de tableau pour décrire les paramètres de composants des primitives de service d'ASE. Les paramètres qui s'appliquent à chaque groupe de primitives de service sont consignés en tableaux. Chaque tableau comporte jusqu'à cinq colonnes:

a) nom de paramètre

b) primitive "request" (demande)

c) primitive "indication";

d) primitive "response" (réponse)

e) primitive "confirmation".

Un paramètre, ou un composant, est énuméré dans chaque rangée de chaque tableau. Dans les colonnes appropriées de la primitive de service, un code est utilisé pour spécifier le type d'usage du paramètre sur la primitive spécifiée dans la colonne:

M Le paramètre est obligatoire pour la primitive.

U Le paramètre est une option de l'utilisateur et peut ou peut ne pas être fourni, cela dépendant de l'usage dynamique de l'utilisateur du service. Lorsqu'il n'est pas fourni, une valeur par défaut est supposée pour le paramètre;

C Le paramètre est conditionné à d'autres paramètres ou à l'environnement de l'utilisateur du service;

— (blanc/vide) Le paramètre n'est jamais présent.

S Le paramètre est un élément sélectionné.

Certaines entrées sont en plus qualifiées par des éléments entre parenthèses. Ceux-ci peuvent être:

a) une contrainte spécifique au paramètre:

"(=)" indique que le paramètre équivaut du point de vue de la sémantique au paramètre dans la primitive de service située immédiatement à sa gauche dans le tableau;

b) une indication qu'une certaine note s'applique à l'entrée;

"(n)" indique que la note "n" suivante contient des informations complémentaires relatives au paramètre et à son utilisation.

3.5.4.3 Procédures de service

Ces procédures de service sont définies en termes des:

a) interactions entre entités d'application par le biais de l'échange d'unités APDU de bus de terrain;

b) interactions entre un fournisseur de services de couche application et un utilisateur de service de couche application dans le même système via l'invocation de primitives de service de couche application.

Ces procédures sont applicables à des instances de communication entre systèmes qui prennent en charge des services de communications à contrainte temporelle au sein de la couche application du bus de terrain.

4 Concepts

4.1 Concepts communs

4.1.1 Vue d'ensemble

Le bus de terrain est destiné à être utilisé dans des usines et centrales de traitement pour interconnecter des dispositifs d'automatisation primaires (par exemple: capteurs, actionneurs, dispositifs locaux d'affichage, annonceurs, automates programmables, petits contrôleurs à boucle unique, et commandes autonomes de terrain) avec des équipements de commande et de surveillance situés dans des salles de contrôle.

Les dispositifs d'automatisation primaires sont associés aux niveaux les plus bas de la hiérarchie d'automatisation industrielle et accomplissent un jeu limité de fonctions dans les limites d'une fenêtre temporelle définie. Certaines de ces fonctions incluent le diagnostic, la validation de données, et la gestion de multiples entrées et sorties.

Ces dispositifs d'automatisation primaires, également appelés "dispositifs de terrain", sont placés à proximité des fluides de processus, de la partie fabriquée, de la machine, de l'opérateur, et de l'environnement. Cet usage positionne le bus de terrain aux niveaux les plus bas de l'architecture de la productique (CIM[8]).

Quelques-uns des avantages prévus de l'utilisation des systèmes de bus de terrain sont les réductions des câblages, l'augmentation de la quantité de données échangées, une plus large distribution de la commande entre les dispositifs d'automatisation primaires et les équipements de salle de contrôle, et la satisfaction aux contraintes à temps critique.

Le présent paragraphe décrit les principes fondamentaux de la FAL. Des informations descriptives détaillées relatives à chacun des ASE de la FAL peuvent être consultées dans le paragraphe "Vue d'ensemble" de chacune des spécifications de modèle de communication.

4.1.2 Relations architecturales

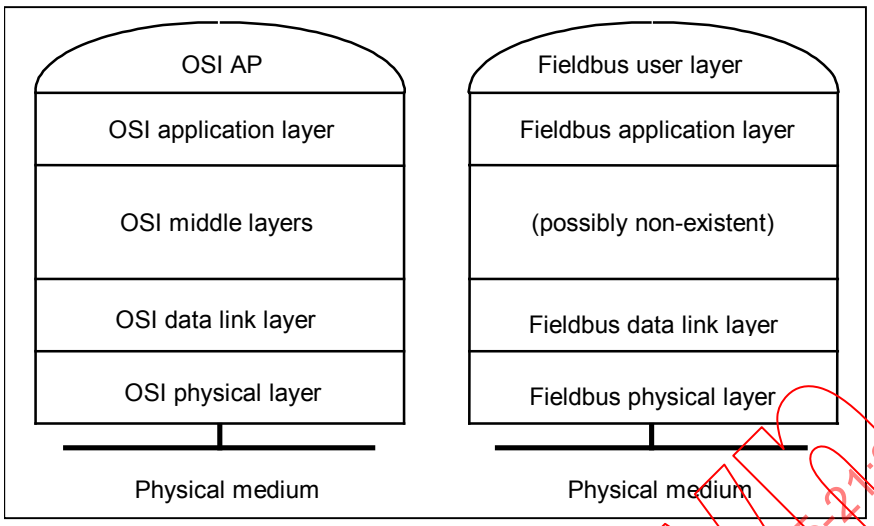
4.1.2.1 Relation à la couche application du modèle de référence de base de l'OSI

Les fonctions de la FAL ont été décrites selon les principes de stratification de l'OSI. Cependant, la relation architecturale de la FAL aux couches inférieures est différente, comme suit (voir Figure 1):

a) la FAL regroupe des fonctions de l'OSI avec des extensions pour couvrir des exigences à temps critique. La norme relative à la structure de la couche application de l'OSI (ISO/CEI 9545) a été utilisée comme base pour la spécification de la FAL;

b) la FAL utilise directement les services de la couche sous-jacente. La couche sous-jacente peut être la DLL ou n'importe quelle couche intermédiaire. Lors de l'utilisation de la couche sous-jacente, la FAL peut fournir des fonctions qui sont normalement associées aux couches intermédiaires de l'OSI pour une mise en correspondance correcte à la couche sous-jacente.

8 CIM = *Computer Integrated Manufacturing*



Légende

Anglais	Français
(possibly non-existent)	(éventuellement absentes)
Fieldbus application layer	Couche application de bus de terrain
Fieldbus data link layer	Couche liaison de données de bus de terrain
Fieldbus physical layer	Couche physique de bus de terrain
Fieldbus user layer	Couche utilisateur de bus de terrain
OSI AP	Processus d'application OSI
OSI application layer	Couche application OSI
OSI data link layer	Couche liaison de données OSI
OSI middle layers	Couches intermédiaires OSI
OSI physical layer	Couche physique OSI
Physical medium	Support physique

Figure 1 – Relation au Modèle de référence de base de l'OSI

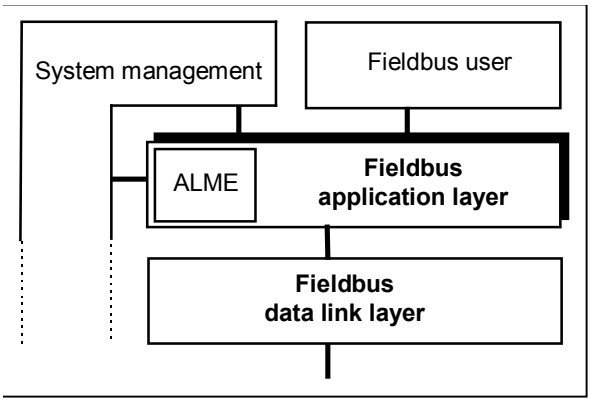
4.1.2.2 Relations à d'autres entités de bus de terrain

4.1.2.2.1 Généralités

Les relations architecturales de la FAL illustrées à la Figure 2 ont été conçues pour prendre en charge les besoins d'interopérabilité de systèmes à temps critique répartis dans l'environnement des bus de terrain.

Dans cet environnement, la FAL fournit des services de communication à des applications à temps critique et à temps non critique situées dans les dispositifs de bus de terrain.

En plus, la FAL utilise directement la DLL pour transférer ses unités de données de protocole de couche application, avec l'aide d'un jeu de services de transfert et d'un jeu de services d'appui pour maîtriser les aspects opérationnels de la DLL.



Légende

Anglais	Français
Fieldbus user	Utilisateur de bus de terrain
Fieldbus application layer	Couche application de bus de terrain
ALME	Application Layer Management Entity (Entité de gestion de couche application)
Fieldbus data link layer	Couche liaison de données de bus de terrain
System management	Gestion de système

Figure 2 – Positionnement architectural de la couche application des bus de terrain

4.1.2.2.2 Utilisation de la couche liaison de données des bus de terrain

La FAL fournit l'accès réseau aux AP de bus de terrain. Elle s'interface directement à la DLL de bus de terrain pour le transfert de ses APDU.

La DLL fournit divers types de services à la FAL pour le transfert de données entre les points d'extrémité de liaison de données (par exemple: les DLSAP et les DLCEP).

4.1.2.2.3 Prise en charge d'applications de bus de terrain

Les applications de bus de terrain apparaissent au réseau sous la forme de processus d'application (AP). Les AP sont les composants d'un système distribué qui peuvent être identifiés et accessibles individuellement.

Chaque AP contient une AE de FAL qui fournit l'accès réseau pour l'AP. Autrement dit, chaque AP communique avec les autres AP par l'intermédiaire de son AE. Dans ce sens, l'AE fournit une fenêtre de visibilité dans l'AP.

Les AP contiennent des composants identifiables qui sont également visibles à travers le réseau. Ces composants sont représentés au réseau comme des APO. Ils peuvent être identifiés par un ou plusieurs attributs clés. Ils sont situés à l'adresse du processus d'application qui les contient.

Les services utilisés pour accéder aux APO sont fournis par des éléments de service d'application (ASE) spécifiques à l'APO qui sont contenus dans la FAL. Ces ASE sont conçus pour prendre en charge les applications d'utilisateur, de blocs fonctionnels, et de gestion.

4.1.2.2.4 Prise en charge de la gestion système

Les services de la FAL peuvent être utilisés pour prendre en charge diverses opérations de gestion, y compris la gestion de systèmes de bus de terrain, et le bus de terrain.

4.1.2.2.5 Accès aux entités de gestion de couche de la FAL

Une LME peut être présente dans chaque entité de la FAL sur le réseau. Les entités de gestion de couche application des bus de terrain (FALME, fieldbus application layer management entity) fournissent l'accès à la FAL à des fins de gestion de système.

Le jeu de données accessibles par le gestionnaire de système est appelé Base d'informations de gestion de système (SMIB, system management information base). Chaque FALME fournit la partie FAL de la SMIB. La mise en œuvre de la SMIB ne relève pas du domaine d'application de la présente norme.

4.1.3 Structure de la couche application des bus de terrain

4.1.3.1 Vue d'ensemble

La structure de la FAL est un raffinement de la structure de couche application de l'OSI (ISO/CEI 9545). Par conséquent, l'organisation de ce paragraphe est semblable à celle de l'ISO/CEI 9545. Certains concepts présentés ici ont été affinés par rapport à l'ISO/CEI 9545 pour l'environnement de bus de terrain.

La FAL diffère des autres couches du Modèle de référence de base de l'OSI par les deux aspects principaux suivants:

a) le Modèle de référence de base de l'OSI définit l'association, un type simple de voie de communications de couche application pour relier les AP les uns aux autres. La FAL définit la relation d'applications (AR), dont il existe plusieurs types, pou permettre à des processus d'application (AP) de communiquer les uns avec les autres;

b) pour transférer ses APDU, la FAL utilise la DLL et non la couche de présentation de l'OSI. Par conséquent, il n'y a pas de contexte de présentation explicite disponible pour la FAL. Le protocole de FAL ne peut pas être utilisé concomitamment avec d'autres protocoles de couche application entre la même paire ou le même jeu de points d'accès au service de liaison de données.

4.1.3.2 Concepts fondamentaux

Le fonctionnement des systèmes ouverts réels à temps critique est modélisé en termes d'interactions entre des AP à temps critiques. La FAL permet à ces AP de passer des commandes et des données entre eux.

La coopération entre des AP exige qu'ils partagent des informations suffisantes pour interagir et accomplir des activités de traitement d'un manière coordonnée. Ces activités peuvent être limitées à un seul segment de bus de terrain ou peuvent s'étendre sur plusieurs segments. La FAL a été conçue à l'aide d'une architecture modulaire pour venir à l'appui des exigences de messagerie de ces applications.

La coopération entre les AP exige parfois aussi qu'ils partagent un sens commun du temps. La FAL ou la DLL peut prévoir la distribution du temps à tous les dispositifs. Elles peuvent aussi définir de services de dispositifs locaux qui peuvent être utilisés par les AP pour accéder au temps distribué.

Le reste du présent paragraphe décrit chacun des composants modulaires de l'architecture et leurs relations des uns aux autres. Les composants de la FAL sont modélisés comme des objets, chacun de ceux-ci fournissant un jeu de services de communications de la FAL destinés à être utilisés par les applications. Les objets de la FAL et leurs relations sont décrits ci-après. Les spécifications particulières d'objets de la FAL et de leurs services sont fournies dans les articles ci-après de la présente norme. La CEI 61158-6-21:2010 spécifie les protocoles nécessaires pour acheminer ces services d'objets entre les applications.

4.1.3.3 Processus d'application des bus de terrain

4.1.3.3.1 Définition de l'AP de bus de terrain

Dans l'environnement de bus de terrain, une application peut être partitionnée en un jeu de composants et répartie sur un certain nombre de dispositifs présents sur le réseau. Chacun de ces composants est appelé "AP de bus de terrain". Un AP de bus de terrain est une variante d'AP tel que défini dans le Modèle de référence OSI (ISO/CEI 7498-1) de l'ISO. Les AP de bus de terrain peuvent être accessibles sans ambiguïté par au moins une adresse individuelle de point d'accès au service DLL. L'adressage non ambigu dans ce contexte signifie qu'aucun autre AP ne peut être situé simultanément à la même adresse. Cette définition n'interdit pas qu'un AP soit situé à plus d'une adresse de point d'accès au service de liaison de données (data link service access point, DLSAP), individuelle ou de groupe.

4.1.3.3.2 Services de communication

Les AP de bus de terrain communiquent les uns avec les autres par des services confirmés et non confirmés (ISO/CEI 10731). Les services définis dans la présente norme pour la FAL spécifient la sémantique des services tels que vus par les AP qui demandent et qui répondent. La syntaxe des messages utilisés pour acheminer les demandes de service et les réponses est définie dans la CEI 61158-6-21:2010. Le comportement de l'AP associé aux services est spécifié par l'AP.

Les services confirmés sont utilisés pour définir des échanges de demande/réponse entre les AP.

D'autre part, les services non confirmés sont utilisés pour définir un transfert unidirectionnel de messages d'un AP vers un ou plusieurs AP distants. Du point de vue des communications, il n'y a pas de relation entre les invocations séparées des services non confirmés comme il en existe entre la demande et la réponse relatives à un service confirmé.

4.1.3.3.3 Interactions d'AP

4.1.3.3.3.1 Généralités

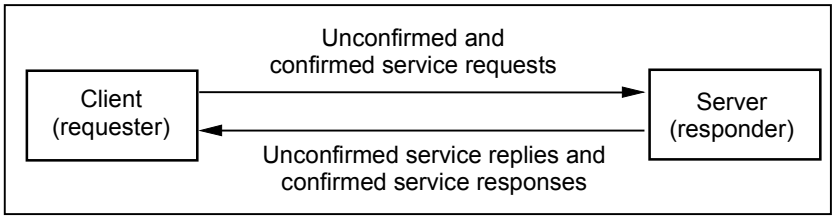
Dans l'environnement de bus de terrain, les AP peuvent interagir avec d'autres AP selon les besoins pour atteindre leurs objectifs fonctionnels. Aucune contrainte n'est imposée par la présente norme sur l'organisation de ces interactions ou les éventuelles relations qui peuvent exister entre elles.

Par exemple, dans l'environnement de bus de terrain, les interactions peuvent être basées sur des messages de demande/réponse envoyés directement entre des AP ou sur des données/événements envoyé(e)s par un AP en vue de leur utilisation par d'autres. Ces deux modèles d'interactions entre les AP sont respectivement appelés "interactions client/serveur" et "interactions publicateur/abonné".

Les services pris en charge par un modèle d'interaction sont acheminés par des points d'extrémité de relation d'applications (AREP) associés aux AP communicants. Le rôle que l'AREP joue dans l'interaction (par exemple: serveur, homologue, publicateur ou abonné) est défini comme un attribut de l'AREP.

4.1.3.3.3.2 Interactions client/serveur

Les interactions client/serveur sont caractérisées par un flot unidirectionnel des données entre un AP client et un ou plusieurs AP serveurs. La Figure 3 illustre l'interaction entre un client seul et un serveur seul. Dans ce type d'interaction, le client peut émettre une demande confirmée ou non confirmée au serveur pour accomplir une certaine tâche. Si le service est confirmé, le serveur retournera toujours une réponse. Si le service est non confirmé, le serveur peut retourner une réponse en utilisant un service non confirmé défini à cette fin.



Légende

Anglais	Français
Client (requester)	Client (demandeur)
Server (responder)	Serveur (répondeur)
Unconfirmed and confirmed service requests	Demandes de services non confirmés et confirmés
Unconfirmed service replies and confirmed service responses	Réponses de services non confirmés et confirmés

Figure 3 – Interactions client/serveur interactions

4.1.3.3.3.3 Interaction publicateur/abonné

4.1.3.3.3.3.1 Généralités

Les interactions publicateur/abonné impliquent un seul AP publicateur et un groupe d'un ou plusieurs AP abonnés. Ce type d'interaction a été définie pour prendre en charge les variantes de deux modèles d'interactions entre des AP: le modèle “pull” et le modèle “push”. Dans ces deux modèles, l'établissement de l'AP publicateur ne relève pas du domaine d'application de la présente norme. Le protocole de Type 21 prend en charge seulement le modèle “push”.

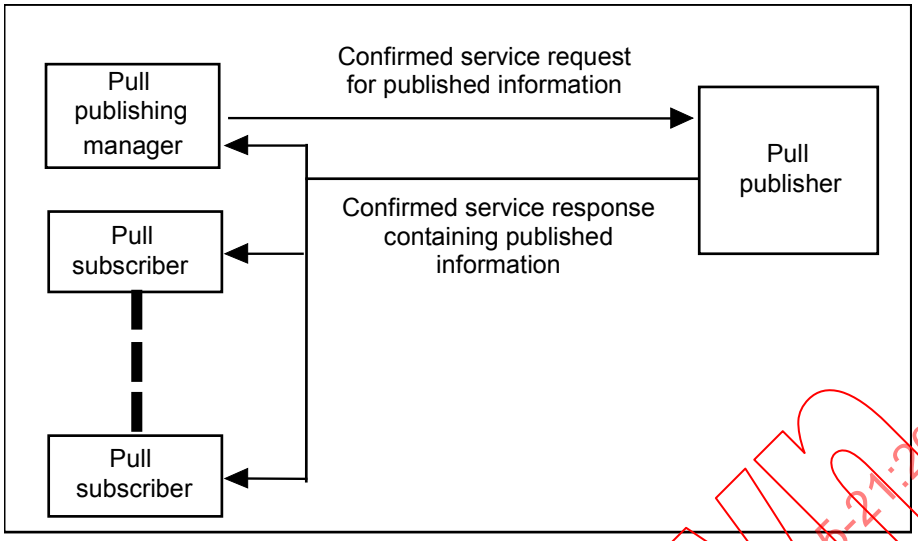
4.1.3.3.3.3.2 Interactions du modèle "pull"

Dans le modèle “pull”, le publicateur reçoit une demande d' provenant d'un gestionnaire d' distant et diffuse (ou multidiffuse) sa réponse sur le réseau. Le gestionnaire d' est chargé seulement de lancer l' en envoyant une demande au publicateur.

Les abonnés souhaitant recevoir les données éditées se mettent à l'écoute pour déceler les réponses émises par le publicateur. De cette façon, les données sont “pulled” (extraites) du publicateur par des demandes issues du gestionnaire d'.

Les services confirmés de la FAL sont utilisés pour prendre en charge ce type d'interaction. Deux caractéristiques de ce type d'interaction le différencient des autres. Premièrement, un échange confirmé de demande/réponse type est accompli entre le gestionnaire d' et le publicateur. Cependant, le mécanisme d'acheminement sous-jacent fourni par la FAL retourne la réponse non seulement au gestionnaire d', mais aussi à tous les abonnés souhaitant recevoir les informations éditées. Cela s'accomplit en faisant transmettre la réponse par la DLL vers une adresse de groupe, plutôt que vers l'adresse individuelle du gestionnaire d'. Par conséquent, la réponse envoyée par le publicateur contient les données éditées et est multidiffusée vers le gestionnaire d' et vers tous les abonnés.

La seconde différence concerne le comportement des abonnés. Les abonnés du modèle "pull", appelés "abonnés "pull"", sont capables de recevoir des données éditées contenues dans des réponses de service confirmé sans avoir émis la demande correspondante. La Figure 4 illustre ces concepts.



Légende

Anglais	Français
Pull publishing manager	Gestionnaire d' «pull»
Pull subscriber	Abonné «pull»
Pull publisher	publicateur «pull »
Confirmed service request for published information	Demande de services confirmés relative à des informations publiées
Confirmed service response containing published information	Réponse de services confirmés contenant des informations publiées

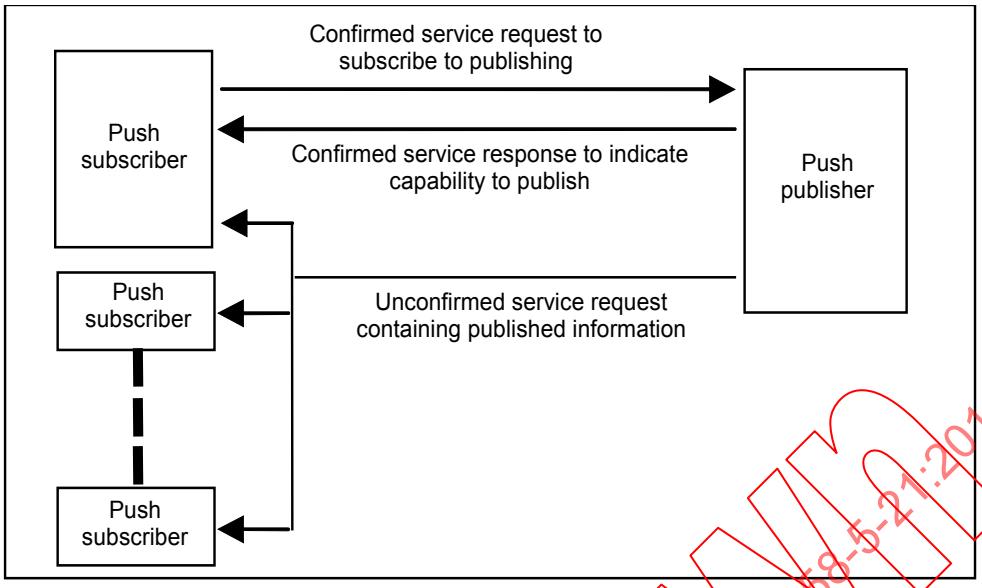
Figure 4 – Interactions du modèle "pull"

4.1.3.3.3.3 Interactions du modèle "push"

Dans le modèle “push”, deux services peuvent être utilisés: l'un confirmé et l'autre non confirmé. Le service confirmé est utilisé par l'abonné dans la demande pour s'abonner à l'activité d'. La réponse à cette demande est retournée à l'abonné, suivant le modèle d'interaction client/serveur. Cet échange est seulement nécessaire lorsque l'abonné et le publicateur sont situés dans des AP différents.

Le service non confirmé dans le modèle "push" est utilisé par le publicateur pour distribuer ses informations aux abonnés. Dans ce cas, le publicateur est chargé d'invoquer le service non confirmé correct en temps opportun et pour fournir des informations appropriées. De cette façon, il est configuré pour “push” (pousser) ses données sur le réseau.

Les abonnés pour le modèle "push" (pousser) reçoivent les services non confirmés publiés qui sont distribués par les publicateurs. La Figure 5 illustre le concept du modèle "push" .



Légende

Anglais	Français
Push subscriber	Abonné «push» "
Push publisher	Publicateur «push»
Confirmed service request to subscribe to publishing	Demande de services confirmés pour s'abonner à la publication
Unconfirmed service request containing published information	Demande de services non confirmés contenant des informations publiées
Confirmed service response to indicate capability to publish	Réponse de services confirmés pour indiquer la capacité de publier

Figure 5 – Interactions du modèle "push"

4.1.3.3.3.4 Actualité des informations éditées

Afin de prendre en charge la nature périssable des informations éditées, la FAL peut prendre en charge quatre types d'actualité définis pour les interactions publicateur/abonné. Chacun permet aux abonnés à des données éditées de déterminer si les données qu'ils reçoivent sont actuelles ou périmées. Ces types sont réalisés par le biais de mécanismes à l'intérieur de la DLL. Chacun de ceux-ci est décrit brièvement au Tableau 1.

Tableau 1 – Types d'actualité

Type	Description
Transparent	Ce type d'actualité permet au processus d'application utilisateur de déterminer la qualité d'actualité des données qu'il génère et de faire que la qualité d'actualité accompagne les informations lorsqu'elles sont transférées à travers le réseau. Dans ce type, le réseau ne fournit ni calcul ni mesure de l'actualité. Il achemine simplement la qualité d'actualité fournie avec les données par le processus d'application utilisateur.
Residence	Lorsque la FAL présente des données issues de l'AP de publication à la DLL en vue de leur émission, la DLL démarre un temporisateur. Si le temporisateur expire avant que les données aient été émises, la DLL marque le tampon comme étant "non opportune" et achemine ces informations d'actualité avec les données.
Synchronized	Ce type d'actualité requiert la coordination de deux éléments d'informations éditées: les données à publier et une "sync mark" (marque de synchronisation) spéciale. Lorsque la "sync mark" est reçue du réseau, un temporisateur démarre dans chacun des postes participants. Par la suite, lorsque des données sont reçues pour émission par la DLL au niveau du poste de publication ou lorsque les données émises sont reçues du réseau au niveau du poste abonné, l'attribut "actualité de DLL" pour les données est mis à TRUE. Il reste à TRUE jusqu'à la réception de la prochaine "sync mark" ou jusqu'à ce que le temporisateur expire. Les données

Type	Description
	reçues après l'expiration du temporisateur, mais avant l'arrivée de la prochaine "sync mark" n'entraînent pas la réinitialisation à TRUE de l'attribut d'actualité. Celui-ci n'est remis à TRUE que si les données sont reçues dans les limites de la fenêtre temporelle après la réception de la "sync mark". Les données émises par le poste de publication avec l'attribut d'actualité mis à FALSE maintiennent le positionnement à FALSE au niveau de chacun des abonnés, quel que soit le fonctionnement de leur temporisateur.
Update	Ce type d'actualité requiert la coordination des mêmes deux éléments d'informations éditées qui sont définis pour l'actualité "synchronized" (synchronisée). Dans ce type, la "sync mark" démarre également un temporisateur dans chacun des postes participants. D'une manière similaire au cas de l'actualité "synchronized", l'expiration du temporisateur force toujours le positionnement à FALSE de l'attribut d'actualité. Contrairement à l'actualité "synchronized", la réception de nouvelles données à tout moment (pas seulement dans les limites de la fenêtre temporelle déclenchée par la réception d'une "sync mark") force le positionnement à TRUE de l'attribut d'actualité.

4.1.3.3.4 Structure d'AP

Les parties internes des AP peuvent être représentées par un ou plusieurs APO et être accessibles par le biais d'une ou plusieurs AE. Les AE fournissent les moyens de communication de l'AP. Pour chaque AP de bus de terrain, il y a une et un seule AE de FAL. Les APO sont la représentation réseau des capacités spécifiques à l'application (objets de processus d'application utilisateur) d'un AP qui sont accessibles par le biais de son AE de FAL.

4.1.3.3.5 Classe d'AP

Une classe d'AP est une définition des attributs et des services d'un AP. La définition de classe normalisée pour les AP est définie dans le présent article. Les classes définies par l'utilisateur peuvent aussi être spécifiées. Les identificateurs de classe, également décrits dans le présent article, sont attribués à partir d'un ensemble réservé à cette fin.

4.1.3.3.6 Type d'AP

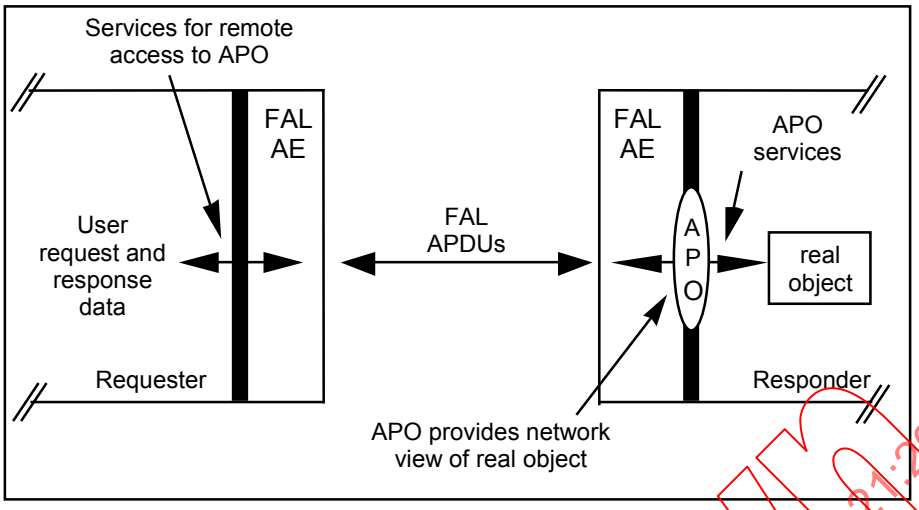
Comme décrit ci-dessus dans les paragraphes précédents, les AP sont définis par l'instanciation d'une classe d'AP. Chaque définition d'AP se compose des attributs et des services qui sont sélectionnés pour l'AP parmi ceux définis par sa classe d'AP. En outre, une définition d'AP contient des valeurs pour un ou plusieurs des attributs sélectionnés pour celui-ci. Lorsque deux AP partagent la même définition, la définition en question est appelée "type d'AP". Ainsi, un type d'AP est une spécification générique d'un AP qui peut être utilisée pour définir un ou plusieurs AP.

4.1.3.4 Objets de processus d'application (APO)

4.1.3.4.1 Définition

Un APO est une représentation réseau d'un aspect spécifique d'un AP. Chaque APO représente un jeu spécifique d'informations et de moyens de traitements d'un AP accessible par le biais de services de la FAL. Les APO sont utilisés pour représenter ces moyens à d'autres AP dans un système de bus de terrain.

Du point de vue de la FAL, un APO est modélisé comme un objet accessible par le réseau qui est contenu dans un AP ou dans un autre APO (les APO peuvent contenir d'autres APO). Les APO fournissent la définition de réseau pour les objets contenus dans un AP qui sont accessibles à distance. La définition d'un APO inclut une identification des services de la FAL qui peuvent être utilisés pour l'accès à distance par des AP distants. Les services de la FAL, tels que montrés à la Figure 6, sont fournis par l'entité de communications de FAL de l'AP, appelée entité d'application de la FAL (FAL AE).



Légende

Anglais	Français
FAL APDUs	Fieldbus Application Layer Application Protocol Data Units (Unités de données de protocole d'application relative à la couche application de bus de terrain) (APDU de FAL)
APO services	Services d'objet de processus d'application
FAL AE	Entité d'application de couche de bus de terrain (AE de FAL)
Responder	Répondeur
Services for remote access to APO	Services pour l'accès distant à l'objet de processus d'application
Requester	Demandeur
APO	Application Process Objet (objet de processus d'application)
real object	objet réel
User request and response data	Données de demande et de réponse utilisateur
APO provides network view of real object	L'APO donne la vue réseau de l'objet réel.

Figure 6 – Services d'APO acheminés par la FAL

Dans la Figure 6, les AP distants agissant comme clients peuvent accéder à l'objet réel en envoyant des demandes par l'intermédiaire de l'APO qui représente l'objet réel. Les aspects locaux de l'AP assurent la conversion entre la vue réseau (l'APO) de l'objet réel et la vue d'AP interne de l'objet réel.

Afin de prendre en charge le modèle d'interaction publicateur/abonné, les informations relatives à l'objet réel peuvent être éditées par l'intermédiaire de son APO. Les AP distants agissant comme abonnés voient la vue APO des informations éditées, au lieu d'avoir à connaître l'un quelconque des détails spécifiques à l'objet réel.

4.1.3.4.2 Classes d'APO

Une classe d'APO est une spécification générique pour un jeu de plusieurs APO, chacun de ceux-ci étant décrit par le même jeu d'attributs et accessible à l'aide du même jeu de services.

Les classes d'APO fournissent le mécanisme pour normaliser des aspects visibles par le réseau des AP. Chaque définition de classe d'APO normalisée spécifie un jeu particulier d'attributs et de services d'AP accessibles par le réseau. La CEI 61158-6-21:2010 spécifie la

syntaxe et les procédures utilisées par le protocole de la FAL pour fournir un accès distant aux attributs et aux services d'une classe d'APO.

Les classes d'APO normalisées sont spécifiées par la présente norme aux fins de normalisation de l'accès distant aux AP. Les classes définies par l'utilisateur peuvent aussi être spécifiées.

Les classes définies par l'utilisateur sont définies comme des sous-classes des classes d'APO normalisées ou comme des sous-classes d'autres classes définies par l'utilisateur. Elles peuvent être définies en identifiant de nouveaux attributs ou en indiquant que les attributs facultatifs pour la classe parant sont obligatoires pour la sous-classe. Les conventions pour définir les classes définies dans le présent article peuvent être utilisées à cette fin. La méthode permettant d'enregistrer ou autrement de rendre ces nouvelles définitions de classe disponibles pour leur utilisation publique ne relève pas du domaine d'application de la présente norme.

4.1.3.4.3 APO en tant qu'instances des classes d'APO

Les classes d'APO sont définies dans la présente norme en utilisant des modèles. Ces modèles sont utilisés non seulement pour définir les classes d'APO, mais aussi pour spécifier les instances d'une classe.

Chaque APO défini pour un AP est une instance d'une classe d'APO. Chaque APO fournit la vue réseau d'un objet réel contenu dans un AP. Un APO est défini en:

a) sélectionnant les attributs dans son modèle de classe d'APO qui doivent être accessibles à partir de l'objet réel;

b) attribuant des valeurs à un ou plusieurs attributs indiqués comme clé dans le modèle. Des attributs clés sont utilisés pour identifier l'APO;

c) en attribuant des valeurs à zéro, un, ou plusieurs attributs non cruciaux pour l'APO. Des attributs non cruciaux sont utilisés pour caractériser l'APO;

d) sélectionnant les services dans le modèle qui peuvent être utilisés par les AP distants pour accéder l'objet réel.

Le paragraphe 3.5.3 spécifie les conventions pour les modèles de classe. Ces conventions assurent la définition des attributs et services obligatoires, facultatifs et conditionnels.

Les attributs et services obligatoires sont tenus d'être présents dans tous les APO de la classe. Les attributs et services facultatifs peuvent être sélectionnés sur une base APO par APO pour leur inclusion dans un APO. Les attributs et services conditionnels sont définis avec un énoncé de contraintes d'accompagnement. Les énoncés de contraintes spécifient les conditions dans lesquelles l'attribut est présent dans un APO.

4.1.3.4.4 Types d'APO

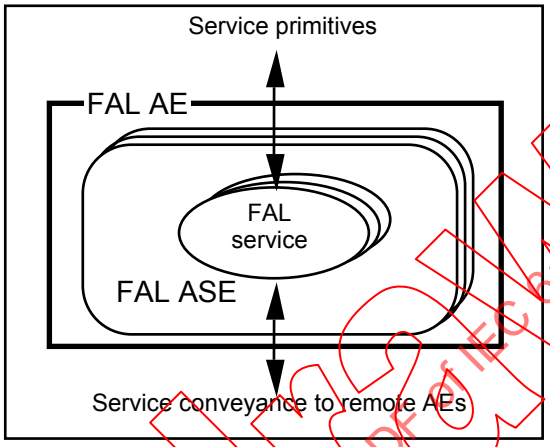
Les types d'APO fournissent le mécanisme pour définir des APO normalisés.

Comme décrit ci-dessus, les APO sont définis en instanciant une classe d'APO. Chaque définition d'APO se compose des attributs et des services qui sont sélectionnés pour l'APO parmi ceux définis par sa classe d'APO. En outre, une définition d'APO contient des valeurs pour un ou plusieurs des attributs sélectionnés pour celui-ci. Lorsque deux APO partagent la même définition à l'exception des positionnements des attributs clés, la définition en question est appelée "type d'APO". Ainsi, un type d'APO est une spécification générique d'un APO qui peut être utilisée pour définir un ou plusieurs APO.

4.1.3.5 Entités d'application

4.1.3.5.1 Définition d'une AE de FAL

Une entité d'application fournit les moyens de communication pour un AP individuel. Une AE de la FAL fournit un jeu de services et les protocoles d'appui pour permettre les communications entre des AP dans un environnement de bus de terrain. Les services fournis par des AE de la FAL sont regroupés en éléments ASE et, de ce fait, les services de FAL fournis à un AP sont définis par les éléments ASE que contient l'AE de la FAL. La Figure 7 illustre ce concept.



Légende

Anglais	Français
FAL ASE	Élément de service d'application de la couche application de bus de terrain (ASE de FAL)
FAL AE	Entité d'application de la couche application de bus de terrain (AE de FAL)
Service primitives	Primitives de services
Service conveyance to remote AEs	Acheminent de service vers les entités d'application distantes
FAL service	Service de la couche application de bus de terrain

Figure 7 – Structure d'une entité d'application

4.1.3.5.2 Type d'AE

Les entités d'application qui fournissent le même jeu d'éléments ASE sont du même type d'AE. Deux AE qui partagent un jeu commun d'éléments ASE sont capables de communiquer l'une avec l'autre.

4.1.3.6 ASE de bus de terrain

4.1.3.6.1 Généralités

Un ASE, tel que défini dans l'ISO/CEI 9545, est un jeu de fonctions d'application qui fournissent un moyen pour l'interfonctionnement des application-entity-invocations ("invocations d'entités d'application) dans un but spécifique. Les ASE fournissent un jeu de services pour acheminer les demandes et réponses à destination et en provenance de processus d'application et de leurs objets. Les AE, telles que définies ci-dessus, sont représentées par un ensemble d'invocations d'ASE au sein de l'AE.

4.1.3.6.2 Services de FAL

Les services de FAL acheminent des demandes/réponses fonctionnelles entre les AP. Chaque service de FAL est définie pour acheminer des demandes et des répondes pour accéder à un objet réel modélisé en tant qu'objet accessible par la FAL.

La FAL définit les services tant confirmés que non confirmés. Les demandes de services confirmés sont envoyées à l'AP contenant l'objet réel. Une invocation d'une demande de service confirmé peut être identifiée par l'ID d'invocation fourni par l'utilisateur. Cet ID d'invocation est retourné dans la réponse par l'AP contenant l'objet réel. Lorsqu'il est présent, il est utilisé par l'AP demandeuse et son AE de FAL pour associer la réponse à la demande appropriée.

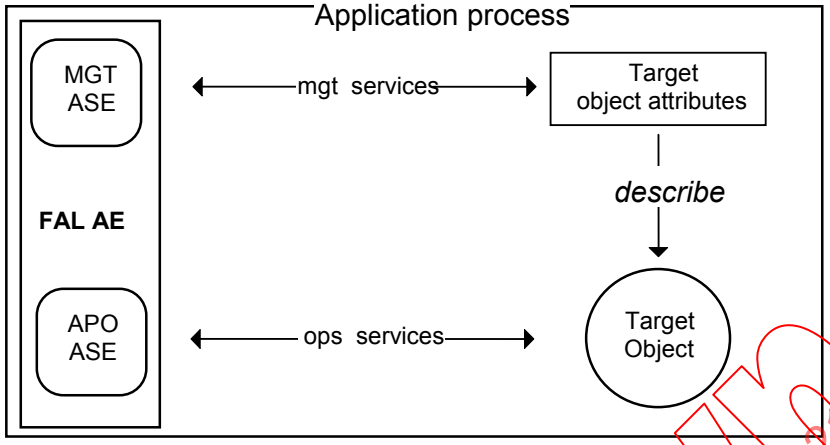
Les services non confirmés peuvent être envoyés de l'AP contenant l'objet réel pour envoyer de l'information relative à l'objet. Ils peuvent aussi être envoyés à l'AP contenant l'objet réel pour accéder à l'objet réel. Les deux types de services non confirmés peuvent être définis pour la FAL.

4.1.3.6.3 Définition des AE de FAL

4.1.3.6.3.1 Généralités

4.1.3.6.3.2 ASE de gestion d'objets

Un ASE spécial de gestion d'objets peut être spécifié pour la FAL afin de fournir des services pour la gestion d'objets. Ses services sont utilisés pour accéder à des attributs d'objets et pour créer et supprimer des instances d'objets. Ces services sont utilisés pour gérer des objets AP visibles du réseau qui sont accessibles à travers la FAL. Les services opérationnels spécifiques qui s'appliquent à chaque type d'objet sont spécifiés dans la définition de l'ASE pour le type d'objet. La Figure 8 illustre l'intégration des services de gestion et des services opérationnels pour un objet au sein d'un AP.



Légende

Anglais	Français
MGT ASE	Élément de service d'application de gestion
APO ASE	Élément de service d'application d'objet de processus application
Target Object	Objet cible
FAL AE	Entité d'application de la couche application de bus de terrain (AE de FAL)
Application process	Processus d'application (AP)
mgt services	service de gestion
ops services	service opérationnel
Target object attributes	Attributs d'objet cible
describe	décrivent

Figure 8 – Gestion d'objets de la FAL

4.1.3.6.3.3 ASE d'AP

Un ASE d'AP peut être spécifié pour l'identification et le contrôle des AP de FAL. Les attributs définis par l'ASE d'AP donnent des détails relatifs au créateur de l'AP et énumèrent son contenu et ses capacités.

4.1.3.6.3.4 ASE d'APO

La FAL spécifie d'un jeu des ASE avec des services définis pour accéder aux APO d'un AP. Les ASE d'APO définis pour la FAL sont définis par chaque modèle de communication.

4.1.3.6.3.5 ASE d'AR

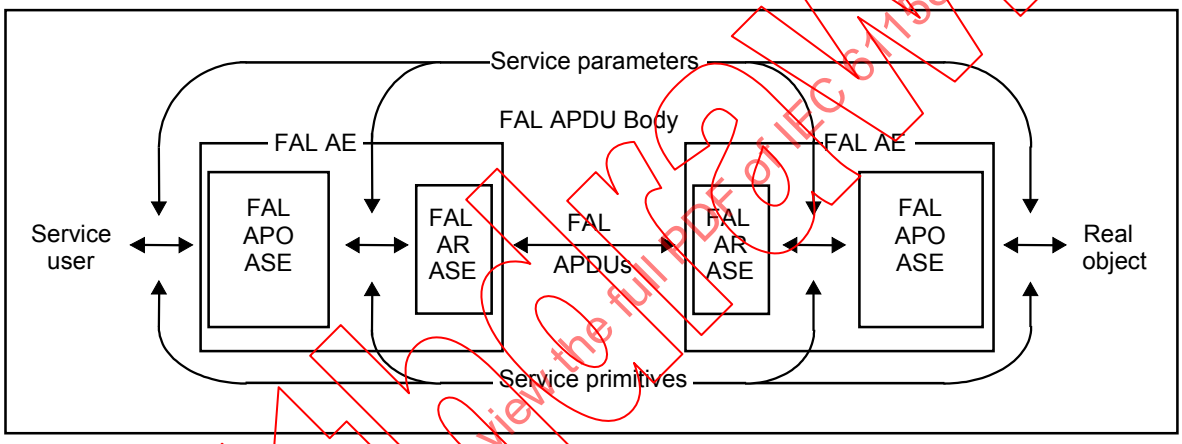
Un ASE d'AR est spécifié pour établir et maintenir des AR, qui sont utilisés pour acheminer des APDU de FAL entre ou parmi des AP. Les AR représentent des voies de communications de couche application entre des AP. Les ASE d'AR sont chargés de fournir des services au niveau des points d'extrémité des AR. Les services d'ASE d'AR peuvent être définis pour établir, cesser et abandonner des AR. Ils peuvent aussi être définis pour acheminer des APDU pour l'AE et pour informer l'utilisateur du statut local de l'AR. En outre, les services locaux peuvent être définis pour accéder à certains aspects des points d'extrémité d'AR.

4.1.3.6.4 Acheminement des services de FAL

Les ASE d'APO de FAL fournissent de services pour acheminer les demandes et les réponses entre les utilisateurs de services et les objets réels.

Pour acheminer les demandes de services et les réponses, trois types d'activités sont définis pour l'utilisateur expéditeur et trois types correspondants sont définis pour l'utilisateur destinataire. Au niveau de l'utilisateur expéditeur, les ASE d'APO de FAL acceptent les demandes de services et les réponses à acheminer. Ils sélectionnent aussi le type d'APDU de FAL qui sera utilisé pour acheminer la demande ou la réponse et codent les paramètres de service dans la partie corps. Ensuite, ils présentent le corps d'APDU codé à l'ASE d'AR en vue de son acheminement.

Au niveau de l'utilisateur destinataire, les ASE d'APO de FAL reçoivent de l'ASE d'AR les corps d'APDU codés. Ils décodent les corps d'APDU et en extraient les paramètres de services que ceux-ci acheminent. Ensuite, ils délivrent la demande de service ou la réponse à l'utilisateur. La Figure 9 illustre ces concepts.



Légende

Anglais	Français
FAL APDUs	Fieldbus Application Layer Application Protocol Data Units (Unités de données de protocole d'application relative à la couche application de bus de terrain) (APDU de FAL)
Service user	Utilisateur du service
Real object	Objet réel
FAL AR ASE	ASE d'AR de la FAL
FAL APO ASE	ASE d'APO de la FAL
FAL APDU Body	Corps d'APDU de la FAL
Service parameters	Paramètres de service
Service primitives	Primitives de services
FAL AE	Entité d'application de la couche application de bus de terrain (AE de FAL)

Figure 9 – Acheminement de services d'ASE

4.1.3.6.5 Contexte de présentation de FAL

Le contexte de présentation dans l'environnement OSI est utilisé pour distinguer les APDU d'un ASE de celles d'un autre et pour identifier les règles de syntaxe de transfert utilisées pour coder chaque APDU. Cependant, l'architecture de communications des bus de terrain

n'inclut pas une couche de présentation. Par conséquent, un mécanisme en variante est fourni pour la FAL par chacun des types spécifiques de modèles de communications.

4.1.3.7 Relations d'applications

4.1.3.7.1 Définition d'une AR

Les AR représentent des voies de communication entre des AP. Elles définissent la manière dont les informations sont communiquées entre les AP. Chaque AR est caractérisée par la manière dont elle achemine les demandes de services d'ASE et les réponses d'un AP vers un autre. Ces caractéristiques sont décrites ci-dessous.

4.1.3.7.2 Points d'extrémité d'AR

Les AR sont définies comme un jeu d'AP coopératifs. L'ASE d'AR dans chaque AP gère un point d'extrémité de l'AR et maintient son contexte local. Le contexte local d'un point d'extrémité d'AR est utilisé par l'ASE de l'AR pour commander l'acheminement des APDU sur l'AR.

4.1.3.7.3 Classes de points d'extrémité d'AR

Les relations d'associations (AR) sont constituées d'un ensemble de points d'extrémité de classes compatibles. Les classes de points d'extrémité d'AR sont utilisées pour représenter les points d'extrémité d'AR qui acheminent les APDU de la même manière. La normalisation des classes de points d'extrémité permet de définir des AR pour différents modèles d'interaction.

4.1.3.7.4 Cardinalité d'AR

Les AR caractérisent les communications entre les AP. L'une des caractéristiques d'une AR est le nombre de points d'extrémité d'AR que contient l'AR. Les AR qui acheminent des services entre deux AP ont une cardinalité de 1:1. Celles qui acheminent des services d'un seul AP vers un certain nombre d'AP ont une cardinalité de 1:many (c'est-à-dire: 1 à plusieurs). Celles qui acheminent des services à destination ou en provenance de plusieurs AP ont une cardinalité de many:many (c'est-à-dire: plusieurs à plusieurs).

4.1.3.7.5 Accès à des objets par l'intermédiaire des AR

Les AR fournissent l'accès aux AP et aux objets qu'ils contiennent par le biais des services d'un ou plusieurs ASE. Par conséquent, une caractéristique est le jeu de services d'ASE qui peuvent être acheminés à destination et en provenance de ces objets par l'AR. La liste des services qui peuvent être acheminés par l'AR est choisie dans ceux définis pour l'AE.

4.1.3.7.6 Trajets d'acheminement d'AR

Les AR sont modélisées comme un ou deux trajets d'acheminement entre des points d'extrémité d'AR. Chaque trajet achemine des APDU dans un sens entre un ou plusieurs points d'extrémité d'AR. Chaque point d'extrémité d'AR destinataire pour un trajet d'acheminement reçoit toutes les APDU émises sur l'AR par le point d'extrémité d'AR expéditeur.

4.1.3.7.7 Rôle d'AREP

Comme les AP interagissent les uns avec les autres par des points d'extrémité, un déterminant fondamental de leur compatibilité est le rôle qu'ils jouent dans l'AR. Le rôle définit la manière dont un AREP interagit avec d'autres AREP dans l'AR.

Par exemple, un AREP peut fonctionner comme client, serveur, publicateur ou abonné. Lorsqu'un AREP interagit avec un autre AREP sur une seule AR à la fois comme client et comme serveur, il est défini pour avoir le rôle de “peer” (c'est-à-dire: homologue).

Certains rôles peuvent être capables de lancer des demandes de services, alors que d'autres peuvent être capables seulement de répondre à des demandes de services. Cette partie de la définition d'un rôle identifie l'exigence pour une AR d'être capable d'acheminer des demandes dans un sens comme dans l'autre ou bien dans un seul sens.

4.1.3.7.8 Tampons et files d'attente d'AREP

Les AREP peuvent être modélisés comme une file d'attente ou comme un tampon ("buffer"). Les APDU transférées sur un AREP placé en file d'attente sont livrées dans l'ordre reçu pour l'acheminement. Le transfert des APDU sur un AREP placé en tampon est différent. Dans ce cas, une APDU devant être acheminée par l'ASE d'AR est placée dans un tampon pour le transfert. Lorsque la DLL obtient l'accès au réseau, elle émet le contenu du tampon.

Lorsque l'ASE d'AR reçoit une autre demande d'acheminement, elle remplace le précédent contenu du tampon, et ce, qu'il ait été émis ou non. Une fois qu'une APDU est écrite dans un tampon pour transfert, elle est conservée dans le tampon jusqu'à ce qu'elle soit écrasée par la prochaine APDU à émettre. Pendant qu'elle est dans le tampon, une APDU peut être lue plus d'une fois sans qu'elle soit effacée du tampon ou sans que son contenu soit modifié.

Le fonctionnement est similaire au niveau de l'extrémité de réception. Le point d'extrémité de réception met une APDU reçue dans un tampon pour qu'elle soit accessible à l'ASE d'AR. Lorsqu'une APDU ultérieure est reçue, elle écrase la précédente APDU dans le tampon, et ce, que celle-ci ait été lue ou non. La lecture de l'APDU dans le tampon n'est pas destructive; elle ne détruit ni ne modifie le contenu du tampon, permettant que le contenu soit lu dans le tampon une ou plusieurs fois.

4.1.3.7.9 Acheminement déclenché par l'utilisateur et programmé

Une autre caractéristique d'un AREP est la trame temporelle dans laquelle il achemine les demandes de services et les réponses. Les AREP déclenchés par l'utilisateur acheminent immédiatement les demandes et les réponses dès leur présentation par l'utilisateur. Ceci est asynchrone par rapport au fonctionnement du réseau.

Un AREP programmé achemine les demandes et les réponses à des intervalles prédéfinis, et ce, quel que soit le moment où elles ont été reçues pour leur transfert. Un AREP programmé peut être capable d'indiquer le moment où les données transférées ont été présentées tard pour l'émission ou le moment où elles ont été présentées à temps, mais émises tard.

4.1.3.7.10 Actualité des AREP

Les AREP acheminent les APDU entre les applications en utilisant les services de la DLL. Lorsque les capacités d'actualité ont été définies pour un AREP et prises en charge par la DLL, l'AREP transmet les indicateurs d'actualité fournis par la DLL. Ces indicateurs permettent aux abonnés à des données éditées de déterminer si les données qu'ils reçoivent sont actuelles ou périmées.

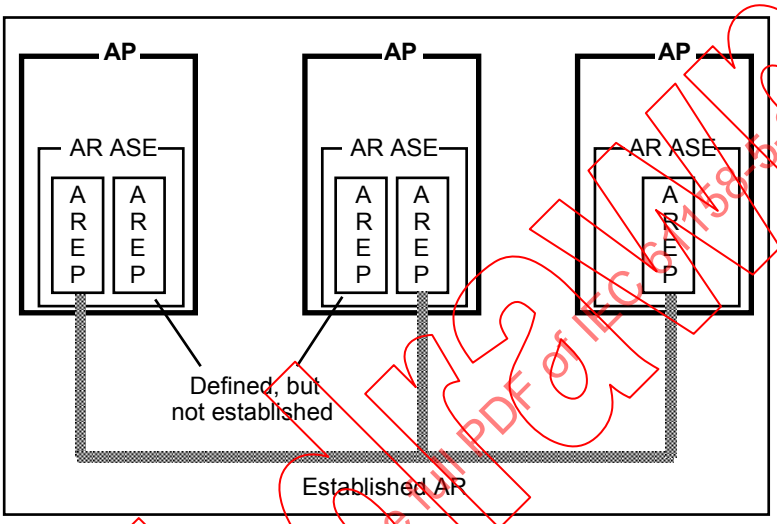
Pour prendre en charge ces types d'actualité, l'AREP publicateur établit une connexion de liaison de données de publicateur reflétant le type d'actualité configuré pour lui par la gestion. Après avoir établi la connexion, l'AREP reçoit les données utilisateur et les présente à la DLL en vue de leur émission lorsque les procédures d'actualité sont exécutées. Lorsque la DLL émet les données, elle inclut l'état courant d'actualité avec les données.

Au niveau de l'AREP abonné, une connexion de ligne de données est ouverte pour recevoir les données éditées qui reflètent le type d'actualité configuré pour lui par la gestion. La DLL calcule l'actualité des données reçues et les envoie ensuite à l'AREP. Les données sont alors délivrées à l'AP utilisateur par le biais de l'ASE approprié.

4.1.3.7.11 Définition et création des AREP

Les définitions d'AREP spécifient les instances des classes d'AREP. Les AREP peuvent être prédéfinis ou peuvent être définis au moyen d'un service “create” si leur AE prend en charge cette fonctionnalité.

Les AREP peuvent être prédéfinis et préétablis ou peuvent être prédéfinis et établis en dynamique. La Figure 10 illustre ces deux cas. Les AREP peuvent également requérir à la fois la définition et l'établissement en dynamique ou peuvent être définis en dynamique de telle manière qu'ils puissent être utilisés sans établissement, c'est-à-dire qu'ils sont définis dans un état établi.



Légende

Anglais	Français
AP	Processus d'application
AR ASE	Élément de service application d'une relation d'applications (ASE d'AR)
AREP	Point d'extrémité d'une relation d'applications
Established AR	Relation d'applications établie
Defined, but not established	Définie, mais pas établie

Figure 10 – AREP définis et établis

4.1.3.7.12 Établissement et fin d'AR

Les AR peuvent être établies soit avant la phase opérationnelle de l'AP, soit pendant le fonctionnement de celui-ci. Lorsqu'elle est établie pendant le fonctionnement d'un AP, l'AR est établie par le biais de l'échange des APDU d'AR.

Une fois qu'une AR a été établie, il peut être mis progressivement fin à l'AR ou elle peut être abandonnée, cela dépendant des fonctionnalités de l'AR.

4.1.4 Désignation et adressage de la couche application des bus de terrain

4.1.4.1 Généralités

Le présent article raffine les principes définis dans l'ISO 7498-3 qui impliquent l'identification (désignation) et la localisation (adressage) des APO référencés à travers la FAL.

Le présent article définit aussi la manière dont les noms et les identificateurs numériques sont utilisés pour identifier les APO accessibles à travers la FAL. En outre, le présent article indique la manière dont les adresses issues des couches sous-jacentes sont utilisées pour localiser les AP dans l'environnement des bus de terrain.

4.1.4.2 Identification des objets accessibles à travers la FAL

4.1.4.2.1 Généralités

Les APO accessibles à travers la FAL sont identifiés indépendamment de leur emplacement. Autrement dit, si l'emplacement de l'AP contenant l'APO change, l'APO peut encore être référencé à l'aide du même jeu d'identificateurs.

Les identificateurs pour les AP et les APO dans la FAL sont définis comme des attributs-clés dans les définitions des classes pour les APO. Deux types d'attribut-clé sont communément utilisés dans ces définitions d'APO: les noms et les identificateurs numériques.

4.1.4.2.2 Noms

Les noms sont des identificateurs orientés chaînes. Ils sont définis pour permettre aux AP et aux APO d'être nommés dans le système où ils sont utilisés. Par conséquent, bien que le domaine d'application d'un nom d'APO soit spécifique à l'AP dans lequel il réside, l'attribution du nom est gérée dans le système dans lequel il est configuré.

Les noms peuvent être descriptifs, mais ce n'est pas obligatoire. Les noms descriptifs permettent de donner des informations sensées relatives à l'objet qu'ils désignent, telles que son utilisation par exemple.

Les noms peuvent également être codés. Les noms codés permettent d'identifier un objet par une courte forme comprimée d'un nom. Ils sont typiquement plus simples à transférer et à traiter, mais plus difficiles à comprendre que les noms descriptifs.

4.1.4.2.3 Identificateurs numériques

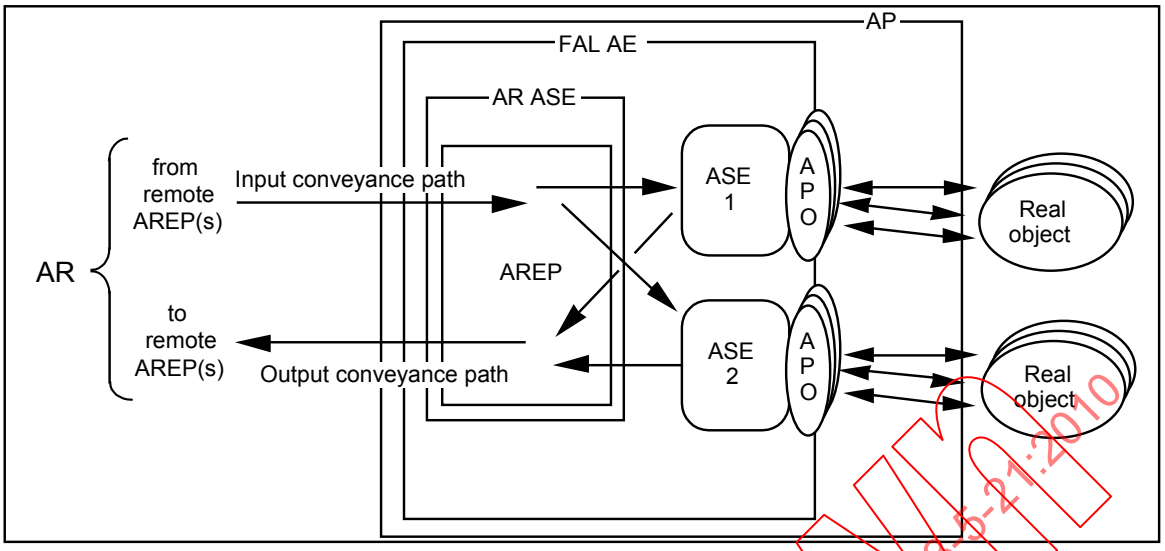
Les identificateurs numériques sont des identificateurs dont les valeurs sont des nombres. Ils sont conçus pour une utilisation efficace dans le système de bus de terrain et peuvent être attribués à des APO par leur AP en vue d'un accès efficace.

4.1.4.3 Adressage des AP accessibles à travers la FAL

Les adresses de bus de terrain représentent les emplacements des AP dans le réseau. Les adresses pertinentes pour la FAL sont les adresses des couches sous-jacentes qui sont utilisées pour localiser les AREP dans un AP.

4.1.5 Résumé de l'architecture

Le présent article propose un résumé de l'architecture de la FAL. La Figure 11 illustre les composants principaux et la manière de les relier le uns aux autres.



Légende

Anglais	Français
AP	Processus d'application
ASE 2	Élément de service d'application 2
ASE 1	Élément de service d'application 12
Real object	Objet réel
AREP	Point d'extrémité d'une relation d'applications
from remote AREP(s)	en provenance d'un ou plusieurs points d'extrémité de relation d'applications
to remote AREP(s)	à destination d'un ou plusieurs points d'extrémité de relation d'applications
FAL AE	Entité d'application de la couche application de bus de terrain (AE de FAL)
APO	Objet de processus d'application
AR ASE	Élément de service d'application d'une relation d'applications
Output conveyance path	Trajet d'acheminement en sortie
Input conveyance path	Trajet d'acheminement en entrée
AR	Relation d'applications

Figure 11 – Composants architecturaux de la FAL

La Figure 11 montre un AP qui communique par l'intermédiaire de l'AE de la FAL. L'AP représente ses objets réels internes comme des APO pour un accès distant à ceux-ci. Il y est montré deux ASE qui fournissent à leurs APO connexes les services d'accès à distance. L'ASE d'AR contient un seul AREP qui achemine les demandes de services et les réponses pour les ASE vers un ou plusieurs AREP distants situés dans des AP distants.

4.1.6 Procédures des services de FAL

4.1.6.1 Procédures des services confirmés de la FAL

L'utilisateur demandeur invoque une primitive de demande de services confirmés de sa FAL. L'ASE approprié de la FAL construit le corps correspondant d'APDU de demande de services confirmés et l'envoie sur l'AR spécifiée.

À la réception du corps d'APDU de demande de services confirmés, l'ASE de réponse le décode. S'il ne s'est pas produit d'erreur de protocole, l'ASE délivre à son utilisateur une primitive "confirmed-service indication" (d'indication de services confirmés).

Si l'utilisateur répondeur est capable de traiter la demande avec succès, l'utilisateur retourne une primitive "confirmed-service response (+)" (c'est-à-dire: réponse positive à une demande de services confirmés).

Si l'utilisateur répondeur est incapable de traiter la demande avec succès, le service échoue et l'utilisateur émet une primitive "confirmed-service response (-)" (c'est-à-dire: réponse négative à une demande de services confirmés) indiquant la cause.

L'ASE répondeur construit un corps d'APDU de "confirmed-service-response" pour une primitive " confirmed-service response (+)" ou un corps d'APDU de "confirmed-service-error" pour une primitive " confirmed-service response (-) " et l'achemine sur l'AR spécifiée.

à la réception du corps d'APDU retourné, l'ASE déclencheur délivre à son utilisateur une primitive "confirmed-service confirmation'" (c'est-à-dire: confirmation de services confirmés) qui spécifie le succès ou l'échec, et la cause de l'échec si échec il y a eu.

4.1.6.2 Procédures des services non confirmés de la FAL

L'utilisateur demandeur invoque une primitive "unconfirmed-service request" (c'est-à-dire: demande de services confirmés) auprès de son AE de FAL. L'ASE approprié de la FAL construit le corps correspondant d'APDU de demande de services non confirmés et l'envoie sur l'AR spécifiée.

À la réception du corps d'APDU de demande non confirmée, l'/les ASE destinataire(s) participant à l'AR délivre(nt) à son/leur utilisateur la primitive "unconfirmed-service indication" (c'est-à-dire: indication de services non confirmés) appropriée. Les paramètres d'actualité sont inclus dans la primitive "indication" si l'AR qui a acheminé le corps d'APDU prend en charge l'actualité.

4.1.7 Attributs communs de la FAL

Dans les spécifications des classes de FAL qui suivent, plusieurs classes utilisent les attributs communs ci-après. Par conséquent, ces attributs sont définis une seule fois ici, au lieu de l'être avec les autres attributs pour chaque classe, à l'exception de la classe de type de données.

ATTRIBUTS:

1	(o)	Attribut clé:	Identificateur numérique
2	(o)	Attribut clé:	Nom
3	(o)	Attribut:	Description d'utilisateur
4	(o)	Attribut:	Révision d'objet

Identificateur numérique
attribut-clé facultatif qui spécifie l'ID numérique de l'objet. Il est utilisé comme une référence sténographique par le protocole de FAL pour identifier l'objet. Il y a trois possibilités pour les besoins d'identification: identificateur numérique, nom, ou les deux. Cet attribut est requis pour le modèle de type de données

Nom
attribut-clé facultatif qui spécifie le nom de l'objet. Il y a trois possibilités pour les besoins d'identification: identificateur numérique, nom, ou les deux

Description d'utilisateur
attribut facultatif qui contient des informations descriptives définies par l'utilisateur relatives à l'objet

Révision d'objet
attribut-clé facultatif qui spécifie le niveau de révision de l'objet. Il est un attribut structuré qui se compose de numéros de révision majeure et mineure. Si la révision d'objet est prise en charge, il contient à la fois une révision majeure et une révision mineure avec une valeur située dans la plage de 0 à 15 pour chacune. Les champs révision majeure et révision mineure sont utilisées comme suit:

a) le champ révision majeure contient la valeur de révision majeure pour l'objet. Une modification apportée à la révision majeure indique que l'interopérabilité est altérée par la modification;

b) le champ révision mineure contient la valeur de révision mineure pour l'objet. Une modification apportée à la révision mineure indique que l'interopérabilité n'a pas été altérée par la modification, c'est-à-dire que les utilisateurs de l'objet continueront de pouvoir interfonctionner avec l'objet si sa révision mineure a changé, à condition que la version majeure reste la même

4.1.8 Paramètres communs aux services de la FAL

Dans les spécifications des services de FAL qui suivent, plusieurs services utilisent les paramètres ci-après. Par conséquent, ils sont définis une seule fois ici, au lieu de l'être avec les autres paramètres pour chacun des services.

AREP
paramètre qui contient des informations suffisantes pour identifier localement l'AREP et qui, de ce fait, peut être utilisé pour acheminer le service. Ce paramètre peut utiliser un attribut-clé de l'AREP pour identifier la relation d'applications. Lorsqu'un AREP prend en charge simultanément plusieurs contextes (établis par le service "initiate"), le paramètre AREP est étendu afin d'identifier le contexte ainsi que l'AREP

Invoke ID (c'est-à-dire: ID d'invocation)
paramètre qui identifie cette invocation du service. Il est utilisé pour associer des demandes de services à des réponses. Par conséquent, deux quelconques invocations de services en cours ne peuvent pas être identifiées par la même valeur de l'ID d'invocation

ASE de FAL/Classe de FAL
paramètre qui spécifie l'ASE de FAL (par exemple: AP, AR, variable, type de donnée, événement, invocation de fonction, région de charge) et la classe de FAL de l'ASE (par exemple: AREP, liste de variables, notificateur, action)

Numeric ID (c'est-à-dire: identificateur numérique)
paramètre qui est l'identificateur numérique de l'objet

Error info
paramètre qui donne des informations d'erreur pour les erreurs de service Il est retourné dans les primitives "confirmed service response (-)". Il se compose des éléments suivants:

a) **Error class** (c'est-à-dire: Classe d'erreur). Ce paramètre indique la classe générale d'erreur. Les valeurs valides sont spécifiées dans la définition du paramètre "error code" (code d'erreur) ci-dessous;

b) **Error code**. Ce paramètre identifie l'erreur de service spécifique;

c) **Additional code** (c'est-à-dire: code supplémentaire). Si une erreur se produit lors du traitement de la demande, ce paramètre facultatif identifie l'erreur spécifique à l'objet auquel l'accès est effectué. Lorsqu'il est utilisé, la valeur présentée dans la primitive "response" est délivrée inchangée dans la primitive "confirmation";

d) **Additional detail** (c'est-à-dire: détail complémentaire). Ce paramètre facultatif contient les données utilisateur qui accompagnent une réponse négative. Lorsqu'il est utilisé, la valeur présentée dans la primitive "response" est délivrée inchangée dans la primitive "confirmation"

4.1.9 Taille des APDU

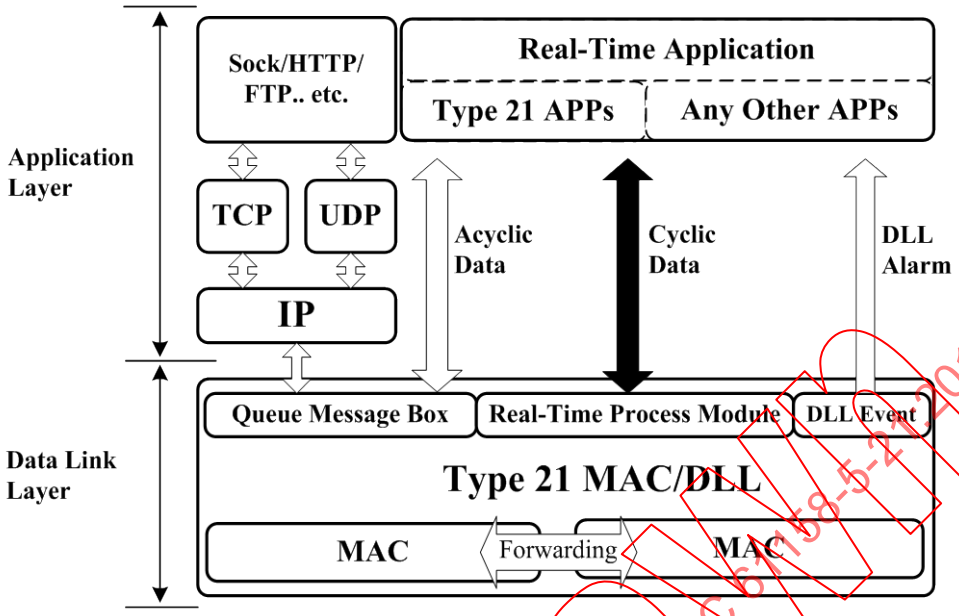
La taille des APDU dépend du modèle de communication.

4.2 Concepts spécifiques de type

Le système d'automation et de contrôle de processus industriels se compose de dispositifs d'automatisation primaires (par exemple: capteurs, actionneurs, dispositifs locaux d'affichage, annonceurs, automates programmables, petits contrôleurs à boucle unique, et commandes autonomes de terrain) avec des équipements de commande et de surveillance

Le transfert de données entre ces dispositifs est accompli par une communication entre homologues ou une communication multidiffusée.

La Figure 12 illustre l'interaction entre la FAL de Type 21 et la DLL. Le Type 21 prend en charge le transfert cyclique et acyclique de données pour ses propres processus d'application. Le Type 21 peut également être utilisé en parallèle avec la communication TCP/IP ou UDP. L'utilisation d'autres protocoles de communication normalisés ne relève pas du domaine d'application de la présente norme.

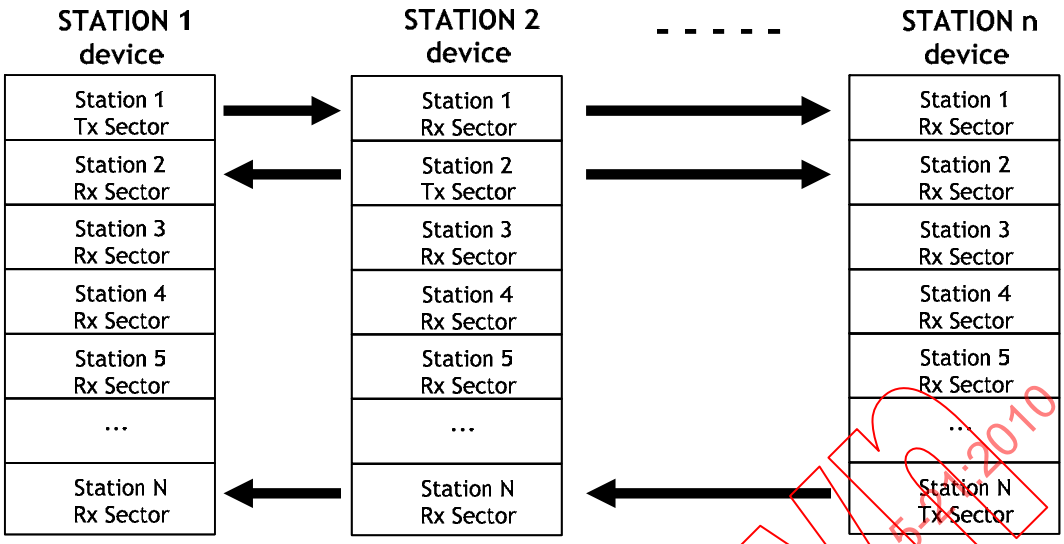


Légende

Anglais	Français
Application Layer	Couche application
Data Link Layer	Couche liaison de données
Sock/HTTP/FTP, etc.	Sock/HTTP/FTP, etc.
Real-time application	Application temps réel
Type 21 APPs	Applications de type 21
Any other APPs	Toutes autres applications
Acyclic data	Données acycliques
Cyclic data	Données cycliques
DLL alarm	Alarme de couche liaison de données
Queue message box	Boîte de messages en file d'attente
Real-time process module	Module de traitement temps réel
DLL event	Événement de couche liaison de données
Type 21 MAC/DLL	MAC/DLL (couche de commande d'accès au support/couche liaison de données) de type 21
MAC	Commande d'accès au support
Forwarding	Réacheminement
TCP	TCP (Protocole de contrôle de transmission)
IP	IP (Protocole Internet)
UDP	UDP (Protocole de datagramme utilisateur)

Figure 12 – Interaction entre FAL et DLL

Le Type 21 prend en charge le modèle de communication publicateur-abonné pour le partage cyclique de données. La Figure 13 illustre ce modèle. Le publicateur multidiffuse périodiquement les données préconfigurées et les abonnés reçoivent les données. Le partage cyclique de données est le modèle le plus largement utilisé dans les applications industrielles.

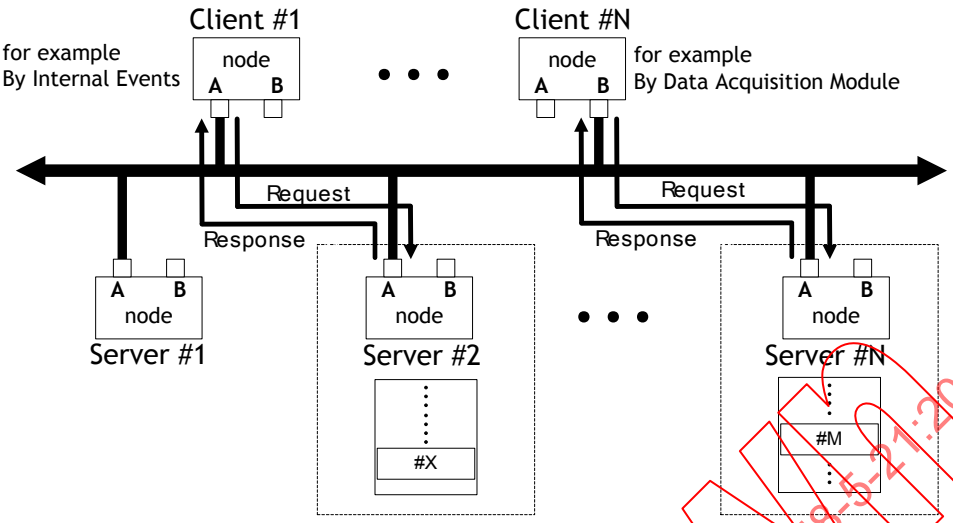


Légende

Anglais	Français
Station 1 device	Dispositif du Poste 1
Station 2 device	Dispositif du Poste 2
Station n device	Dispositif du Poste n
Station 1 Tx sector	Secteur d'émission du Poste 1
Station 1 Rx sector	Secteur de réception du Poste 1
Station N Tx sector	Secteur d'émission du Poste N
Station N Rx sector	Secteur de réception du Poste N

Figure 13 – Modèle de communication publicateur-abonné

Le Type 21 prend en charge le modèle de communication client-serveur pour le transfert événementiel de données La Figure 14 illustre ce modèle. Le client demande des données telles que dictées par des événements internes ou externes. Le serveur répond par les données. Cela peut être utilisé pour les processus d'applications déclenchés par des événements ou déclenchés par l'utilisateur.



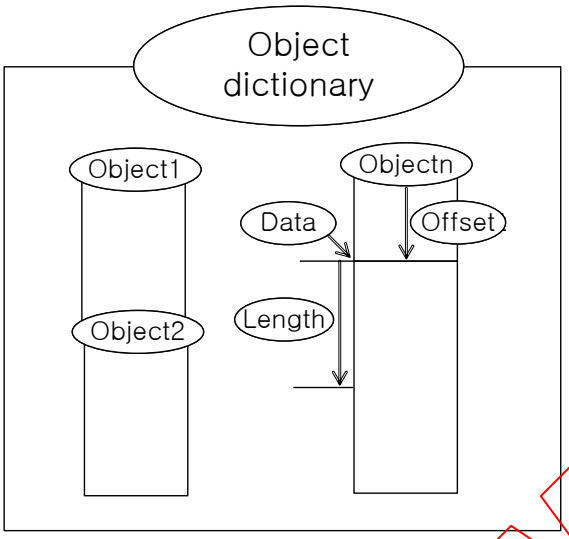
Légende

Anglais	Français
for example by Internal Events	par exemple: par événements internes
Client # 1	Client n° 1
Client # N	Client n° N
node	nœud
for example by Data Acquisition Module	par exemple: par Module d'acquisition de données
Request	Demande
Response	Réponse
Server #1	Serveur n° 1
Server #2	Serveur n° 2
Server #N	Serveur n° N

Figure 14 – Modèle de communication client-serveur

4.2.1 Nœud, AP et dictionnaire d'objets

Chaque nœud héberge exactement un seul AP. Tous les APO pour cet AP sont rassemblés dans le dictionnaire d'objets (OD, object dictionary). La Figure 15 illustre le modèle d'objets.



Légende

Anglais	Français
Object dictionary	Dictionnaire d'objets
Object1	Objet1
Object2	Objet2
Objectn	Objetn
Offset	Décalage
Length	Longueur
Data	Données

Figure 15 – Modèle d'objets

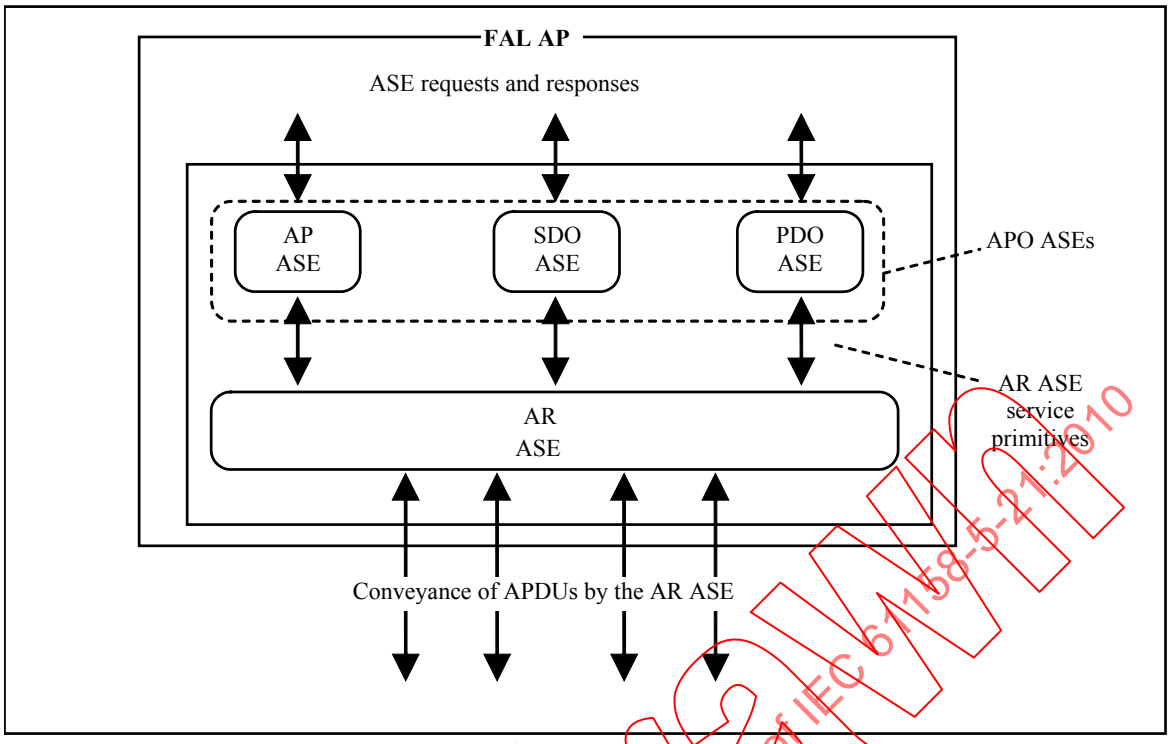
La structure globale de l'OD est telle que décrite au Tableau 2.

Tableau 2 – Structure globale de l'OD

Zone	Contenu
Zone des types de données	Définition des types de données
Zone des profils de communication	Contient les paramètres spécifiques aux communications pour le réseau de Type 21. Ces entrées sont communes à tous les dispositifs
Zone spécifique au fabricant	Définition des variables spécifiques au fabricant
Zone des profils de dispositifs	Définition des variables définies dans le profil de dispositif (hors du domaine d'application du présent document)
Zone réservée	Réservée pour un usage futur

4.2.2 ASE d'APO

Les ASE de FAL d'une application de Type 21 et leurs interrelations sont décrits à la Figure 16.



Légende

Anglais	Français
AR ASE	Élément de service application d'une relation d'applications (ASE d'AR)
ASE requests and responses	Demande et réponse d'élément de service d'application
FAL AP	Processus d'application de la couche application de bus de terrain (AP de FAL)
AP ASE	Élément de service application d'un processus d'application (ASE d'AP)
SDO ASE	Élément de service application d'un objet de données de service (ASE de SDO)
PDO ASE	Élément de service application d'un objet de données de processus (ASE de PDO)
Conveyance of APDUs by the AR ASE	Acheminent des APDU par l'ASE d'AR
AR ASE service primitives	Primitives de services de l'ASE d'AR
APO ASEs	Éléments de service application d'un objet de processus d'application (ASE d'APO)

Figure 16 – ASE d'une application de Type 21

5 Data type ASE

5.1 Généralités

5.1.1 Vue d'ensemble

Les types de données de bus de terrain spécifient la syntaxe indépendante de toute machine pour les données d'application acheminées par les services de FAL. La FAL prend en charge la définition et le transfert des types de données tant de base que construits. Les règles de codage pour les types de données spécifiés dans le présent paragraphe sont données dans la CEI 61158-6-21:2010.

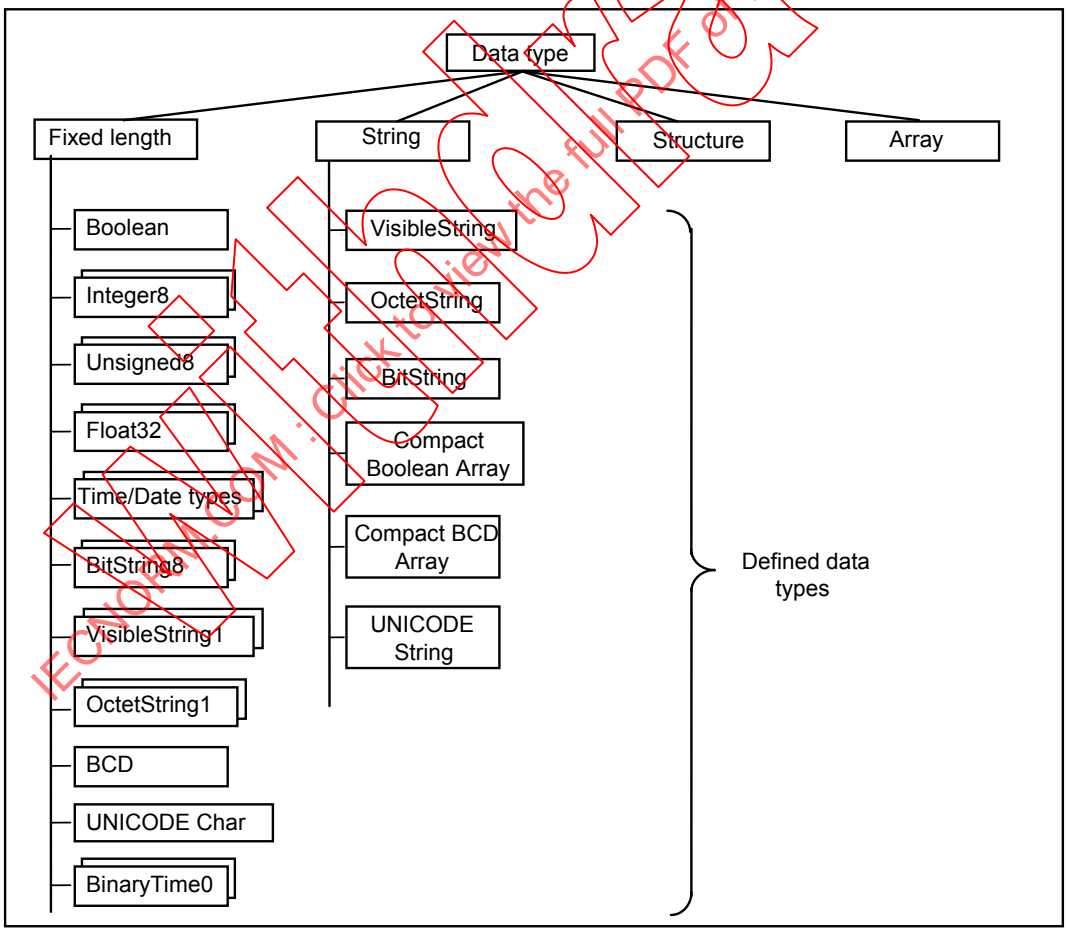
Les types de base sont des types atomiques qui ne peuvent pas être décomposés. Les types construits sont des types composés de types de base et d'autres types construits. La présente norme ne contraint ni leur complexité ni leur profondeur d'imbrication.

Les types de données sont définis comme des instances de la classe de types de données, comme montré à la Figure 17. Seul un sous-ensemble des types de données définis dans le présent article est montré dans cette figure. La définition de nouveaux types est accomplie en fournissant un ID numérique et en fournissant des valeurs pour les attributs définis pour la classe de types de données.

Les définitions des types de données montrées à la Figure 17 sont représentées sous la forme d'une structure classe/format/instance commençant par la classe de types de données intitulée "Data type".

Les classes de données de base sont utilisées pour définir des types de données de longueur fixe et de chaîne de bits. Les types normalisés pris dans l'ISO/CEI 8824 sont appelés types de données simples. Les autres types de données de base normalisés sont définis spécifiquement pour les applications de bus de terrain et sont appelés "types spécifiques".

Les types construits spécifiés dans la présente norme sont des chaînes, des matrices et des structures. Il n'y a pas de types normalisés définis pour les matrices ou les structures.



Légende

Anglais	Français
Defined data types	Types de données définis

Figure 17 – Exemple de hiérarchie de la classe de types de données "Data type"

5.1.2 Vue d'ensemble de types de base

La plupart des types de base sont définis à partir d'un ensemble de types de l'ISO/CEI 8824 (types simples). Certains types de l'ISO/CEI 8824 ont été étendus pour une utilisation spécifique dans le bus de terrain (types spécifiques).

Les types simples sont des types universels de l'ISO/CEI 8824. Ils sont définis dans la présente norme pour leur fournir des identificateurs de classe de bus de terrain.

Les types spécifiques sont des types de base définis spécifiquement pour être utilisés dans l'environnement de bus de terrain. Ils sont définis comme des sous-types de classe simples.

Les types de base ont une longueur constante. Deux variantes sont définies, l'une pour définir les types de données dont la longueur est un nombre entier d'octets et l'autre pour définir les types de données dont la longueur est un nombre arbitraire de bits.

NOTE Boolean, Integer, OctetString, VisibleString, et UniversalTime sont définis dans la présente norme dans le but de leur attribuer des identificateurs de classes de bus de terrain. La présente norme ne change pas leurs définitions telles que spécifiées dans l'ISO/CEI 8824.

5.1.3 Vue d'ensemble des types de longueur fixe

La longueur des types de longueur fixe est un nombre entier d'octets.

5.1.4 Vue d'ensemble des types construits

5.1.4.1 Chaînes

Une chaîne ("string") est composée d'un ensemble ordonné d'un nombre variable d'éléments de longueur fixe typés de façon homogène.

5.1.4.2 Matrices

Une matrice ("array") est composée d'un ensemble ordonné d'éléments typés de façon homogène. La présente norme n'impose pas de restrictions sur le type de données des éléments de matrice, mais exige par contre que chaque élément soit du même type. Une fois qu'il est défini, le nombre d'éléments dans une matrice ne peut pas être modifié.

5.1.4.3 Structures

Une structure est constituée d'un ensemble ordonné d'éléments typés de façon hétérogène appelés "champs". Tout comme dans le cas des matrices, la présente norme ne limite pas le type de données des champs. Cependant, les champs dans une structure peuvent ne pas être du même type.

5.1.4.4 Niveau d'imbrication

La présente norme permet aux matrices et aux structures de contenir d'autres matrices et structures. Elle n'impose aucune restriction au nombre de niveaux d'imbrication autorisés. Cependant, les services de la FAL définis pour accéder à des données prévoient l'accès partiel au niveau négocié par le service "initiate". Le nombre par défaut de niveaux pour l'accès partiel est 1 (un).

Lorsqu'une matrice ou une structure contient des éléments construits, l'accès à un élément pris séparément dans sa totalité est toujours assuré. L'accès à des sous-éléments de l'élément construit est également assuré, mais seulement lorsqu'il est négocié de manière explicite au cours de l'établissement de l'AR ou lorsqu'il est préconfiguré de manière explicite sur des AR préétablies.

NOTE Par exemple, supposons qu'un type de données appelé “employé” soit défini pour contenir la structure "nom d'employé" et que "nom d'employé" soit définie pour contenir "nom de famille" et "prénom". Pour accéder à la structure "employé", la FAL permet l'accès indépendant à la structure entière et au champ de premier niveau “nom d'employé.” Sans négocier de manière explicite l'accès partiel à plus d'un niveau, l'accès indépendant à "nom de famille" ou à " prénom" ne serait pas possible; leurs valeurs pourraient être seulement accessibles ensemble comme un bloc par le biais de l'accès à “employé” ou à “nom d'employé”.

5.1.5 Spécification des types de données définis par l'utilisateur

Les utilisateurs peuvent trouver nécessaire de définir des types de données personnalisés pour leurs propres applications. Les types définis par l'utilisateur sont pris en charge par la présente norme comme des instances des classes de types de données.

Les types définis par l'utilisateur sont spécifiés de la même manière dont tous les objets de FAL sont spécifiés. Ils sont définis en donnant des valeurs aux attributs spécifiés pour leur classe.

5.1.6 Transfert de données d'utilisateur

Les données utilisateur sont transférées entre applications par le protocole de FAL. L'ensemble du codage et du décodage est accompli par l'utilisateur de la FAL.

Les règles pour coder les données d'utilisateur dans les unités de données du protocole de FAL dépendent du type de données. Elles sont définies dans la CEI 61158-6-21:2010. Les types de données définis par l'utilisateur pour lesquels il n'y a pas de règles de codage sont transférés sous la forme de séquences de longueur variable d'octets. Le format des données dans la chaîne d'octets est défini par l'utilisateur.

5.2 Définition formelle des objets de types de données

5.2.1 Classe de types de données "Data type"

5.2.1.1 Modèle

La classe de types de données spécifie la racine de l'arbre de la classe des types de données. Sa classe parent "TOP" indique le sommet de l'arbre de la classe de FAL.

FAL ASE:			**DATA TYPE ASE**
CLASS:			**DATA TYPE**
CLASS ID:			5 (FIXED-LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)
PARENT CLASS:			TOP
ATTRIBUTS:			
1	(o)	Attribut clé:	Data type numeric identifier (Identificateur numérique de type de données)
2	(o)	Attribut clé:	Data type name (Nom de type de données)
3	(m)	Attribut:	Format (FIXED-LENGTH, STRING, STRUCTURE, ARRAY)
4	(c)	Contrainte:	Format = FIXED-LENGTH | STRING
4.1	(m)	Attribut:	Octet length (Longueur en octets)
5	(c)	Contrainte:	Format = STRUCTURE
5.1	(m)	Attribut:	Number of fields (Nombre de champs)
5.2	(m)	Attribut:	List of fields (Liste de champs)
5.2.1	(o)	Attribut:	Field name (Nom de champ)
5.2.2	(m)	Attribut:	Field data type (Type de donnée de champ)

6	(c)	Contrainte:	Format = ARRAY
6.1	(m)	Attribut:	Number of array elements (Nombre d'éléments de la matrice)
6.2	(m)	Attribut:	Array element data type (Type de donnée d'élément de la matrice)

5.2.1.2 Attributs

Data type numeric identifier
attribut facultatif qui identifie l'identificateur numérique du type de données connexe

Data type name
attribut facultatif qui identifie le nom du type de données connexe

Format
attribut qui identifie le type de donnée comme étant fixed-length, string, array, ou data structure (structure de données)

Octet length
attribut conditionnel qui définit la représentation des dimensions de l'objet de type associé. Il est présent lorsque la valeur de l'attribut format est "FIXED-LENGTH" ou "STRING". Pour les types de données FIXED-LENGTH, il représente la longueur en octets. Pour les types de données STRING, il représente la longueur en octets pour un élément simple d'une chaîne

Number of fields
attribut conditionnel qui définit le nombre de champs dans une structure. Il est présent lorsque la valeur de l'attribut format est "STRUCTURE"

List of fields
attribut conditionnel qui est une liste ordonnée de champs contenus dans la structure. Chaque champ est spécifié par son numéro et son type. Les champs sont numérotés séquentiellement à partir de 0 (zéro) dans l'ordre de leur apparition. L'accès partiel à des champs au sein d'une structure est pris en charge en identifiant le champ par son numéro. Cet attribut est présent lorsque la valeur de l'attribut format est "STRUCTURE"

Field name
attribut facultatif conditionnel qui spécifie le nom du champ. Il peut être présent lorsque la valeur de l'attribut format est "STRUCTURE"

Field data type
attribut conditionnel qui spécifie le type de donnée du champ. Il est présent lorsque la valeur de l'attribut format est "STRUCTURE". Cet attribut peut lui-même spécifier un type de donnée construit soit en référençant une définition de type de donnée construit par son ID numérique (c'est-à-dire: identificateur numérique), soit en intégrant ici une définition de type de donnée construit. Pour intégrer une description, il est utilisé la description de type de données intégré qui est montrée ci-après

Number of array elements
attribut conditionnel qui définit le nombre d'éléments de type "array" (matrice). Les éléments de matrices sont indexés de “0” à “*n*-1” pour une matrice de “*n*” éléments. Cet attribut est présent lorsque la valeur de l'attribut format est "ARRAY"

Array element data type
attribut conditionnel qui spécifie le type de donnée pour les éléments d'une matrice. Tous les éléments de la matrice ont le même type de donnée. Il est présent lorsque la valeur de l'attribut format est "ARRAY". Cet attribut peut lui-même spécifier un type de donnée construit soit en référençant une définition de type de donnée construit par son ID numérique (c'est-à-dire: identificateur numérique), soit en intégrant ici une définition de type de donnée construit. Pour intégrer une description, il est utilisé la description de "embedded-data type" (type de données intégré) qui est montrée ci-après

Embedded-data type description (description de type de données intégrées)
attribut qui est utilisé pour définir les types de données intégrés, et ce, de manière récursive dans une structure ou une matrice. Le modèle ci-dessous définit son contenu. Les attributs montrés dans le modèle sont définis ci-dessus dans la classe de types de données, à

l'exception de l'attribut "embedded-data type" (type de données intégré), qui est une référence récursive à cet attribut. Il est utilisé pour définir des éléments imbriqués

ATTRIBUTS:

1	(m)	Attribut:	Format (FIXED-LENGTH, STRING, STRUCTURE, ARRAY)
2	(c)	Contrainte:	Format = FIXED-LENGTH | STRING
2.1	(m)	Attribut:	Data type numeric ID value (Valeur d'identificateur numérique de type de données)
2.2	(m)	Attribut:	Octet length (Longueur en octets)
3	(c)	Contrainte:	Format = STRUCTURE
3.1	(m)	Attribut:	Number of fields (Nombre de champs)
3.2	(m)	Attribut:	List of fields (Liste de champs)
3.2.1	(m)	Attribut:	Embedded data type description (description de type de données intégrées)
4	(c)	Contrainte:	Format = ARRAY
4.1	(m)	Attribut:	Number of array elements (Nombre d'éléments de la matrice)
4.2	(m)	Attribut:	Embedded data type description (description de type de données intégrées)

5.3 Types de données définis pour la FAL

5.3.1 Types Fixed-length (de longueur fixe)

5.3.1.1 Types Boolean (booléens)

CLASS: **Data type** (Type de données)

ATTRIBUTS:

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

Ce type de données exprime le type de données Boolean avec les valeurs TRUE et FALSE.

5.3.1.2 Types pour la date et l'heure

5.3.1.2.1 TimeOfDay

CLASS: **Data type** (Type de données)

ATTRIBUTS:

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

Ce type de données est constitué de deux éléments de valeurs unsigned (non signées) et exprime l'heure du jour et la date. Le premier élément est un type de données Unsigned32 qui donne l'heure après minuit en millisecondes. Le second élément est un type de données Unsigned16 qui donne la date comme un compte de jours à partir de 1984-01-01 (1er janvier 1984).

5.3.1.2.2 TimeDifference

CLASS:		**Data type** (Type de données)	
ATTRIBUTS:			
1	Data type numeric identifier	=	13
2	Data type name	=	TimeDifference
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	6

Ce type de données est constitué de deux éléments de valeur unsigned (non signées) qui expriment une longueur de temps. Le premier élément est un type de données Unsigned32 qui donne une partie fractionnaire d'un jour en millisecondes. Le second élément est un type de données Unsigned16 qui donne le nombre de jours.

5.3.1.3 Types numériques

5.3.1.3.1 Real32

CLASS:		**Type de donnée**	
ATTRIBUTS:			
1	Data type numeric identifier	=	8
2	Data type name	=	Real32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

Ce type a une longueur de quatre octets. Le format de Real32 est celui défini par la CEI 60559 comme étant en simple précision ("single precision").

5.3.1.3.2 Real64

CLASS:		**Type de donnée**	
ATTRIBUTS:			
1	Data type numeric identifier	=	17
2	Data type name	=	Real64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

Ce type a une longueur de huit octets. Le format de Real64 est celui défini par la CEI 60559 comme étant en double précision ("double precision").

5.3.1.3.3 Types entiers

5.3.1.3.3.1 Integer8

CLASS:		**Type de donnée**	
ATTRIBUTS:			
1	Data type numeric identifier	=	2
2	Data type name	=	Integer8
3	Format	=	FIXED-LENGTH
4.1	Octet length	=	1

Ce type entier (Integer) est un nombre binaire en complément à deux avec une longueur égale à un octet.

5.3.1.3.3.2 Integer16

CLASS:		**Data type** (Type de données)	
ATTRIBUTS:			
1	Data type numeric identifier	=	3
2	Data type name	=	Integer16
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	2

Ce type entier (Integer) est un nombre binaire en complément à deux avec une longueur égale à deux octets.

5.3.1.3.3.3 Integer32

CLASS:		**Type de donnée**	
ATTRIBUTS:			
1	Data type numeric identifier	=	4
2	Data type name	=	Integer32
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	4

Ce type entier (Integer) est un nombre binaire en complément à deux avec une longueur égale à quatre octets.

5.3.1.3.3.4 Integer64

CLASS:		**Data type** (Type de données)	
ATTRIBUTS:			
1	Data type numeric identifier	=	21
2	Data type name	=	Integer64
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	8

Ce type entier (Integer) est un nombre binaire en complément à deux avec une longueur égale à huit octets.

5.3.1.3.4 Types non signés (Unsigned)

5.3.1.3.4.1 Unsigned8

CLASS:		**Type de donnée**	
ATTRIBUTS:			
1	Data type numeric identifier	=	5
2	Data type name	=	Unsigned8
3	Format	=	FIXED-LENGTH
3.1	Octet length	=	1

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type non signé (unsigned) a une longueur égale à un octet.