IEC 61158-5-8

Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications –
Part 5-8: Application layer service definition – Type 8 elements

Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 5-8: Définition des services de la couche application – Éléments de Type 8

IEC 61158-5-8:2007-12(en-fr)

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.

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

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 It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

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

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

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

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

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

IEC Just Published - webstore.iec.ch/justpublished
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 15 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary
Plus de 60 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

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

IEC 61158-5-8

Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 5-8: Application layer service definition – Type 8 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-8: Définition des services de la couche application – Éléments de Type 8

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 25.040.40; 35.100.70

ISBN 978-2-8322-2606-3

CONTENTS

FOREWORD....4
INTRODUCTION....6
1 Scope....7
1.1 Overview....7
1.2 Specifications....8
1.3 Conformance....8
2 Normative references....8
3 Terms and definitions....9
3.1 ISO/IEC 7498-1 terms....9
3.2 ISO/IEC 8822 terms....9
3.3 ISO/IEC 9545 terms....9
3.4 ISO/IEC 8824 terms....9
3.5 Fieldbus data-link layer terms....9
3.6 Fieldbus application layer specific definitions....10
3.7 Abbreviations and symbols....16
3.8 Conventions....16
4 Concepts....20
4.1 Overview....20
4.2 Architectural relationships....20
4.3 Fieldbus Application Layer structure....22
4.4 FAL naming and addressing....32
4.5 Architecture summary....33
4.6 FAL service procedures....33
4.7 Notional FAL service procedures [INFORMATIVE]....33
4.8 Common FAL attributes....34
4.9 Common FAL service parameters....35
4.10 APDU size....36
5 Data type ASE....36
5.1 General....36
5.2 Formal definition of data type objects....38
5.3 FAL defined data types....38
5.4 Data type ASE service specification....42
6 Communication model specification....42
6.1 ASEs....42
6.2 ARs....85
6.3 Summary of FAL classes....91
6.4 Permitted FAL services by AREP role....91
Bibliography....93

Figure 1 – Relationship to the OSI basic reference model....20
Figure 2 – Architectural positioning of the Type 8 FAL....21
Figure 3 – Client/server interactions....23
Figure 4 – Push model interactions....24
Figure 5 – APOs services conveyed by the FAL....25
Figure 6 – Application entity structure....27

Figure 7 – Example FAL ASEs 28
Figure 8 – FAL management of objects 29
Figure 9 – ASE service conveyance 30
Figure 10 – Defined and established AREPs 32
Figure 11 – FAL architectural components 33
Figure 12 – Data type class hierarchy example 36
Figure 13 – The AR ASE conveys APDUs between APs 58
Figure 14 – 1-to-1 AR establishment 65

Table 1 – Get attributes service parameters 43
Table 2 – Identify 50
Table 3 – Get status 51
Table 4 – Initiate 53
Table 5 – Terminate 55
Table 6 – Reject 57
Table 7 – Conveyance of service primitives by AREP role 59
Table 8 – Valid combinations of AREP roles involved in an AR 59
Table 9 – AR-unconfirmed send 63
Table 10 – AR-establish service 64
Table 11 – Valid combinations of AREP classes to be related 66
Table 12 – AR-abort 67
Table 13 – AR-Data-Send-Acknowledge service parameters 68
Table 14 – Read service parameters 73
Table 15 – Write service parameters 74
Table 16 – Information report service 75
Table 17 – Start service parameters 79
Table 18 – Stop service parameters 80
Table 19 – Resume service parameters 81
Table 20 – Reset service parameters 82
Table 21 – State transitions for a function invocation object 84
Table 22 – FAL class summary 91
Table 23 – Services by AREP role 92

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-8: Application layer service definition – Type 8 elements

FOREWORD

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

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

International Standard IEC 61158-5-8 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This first edition and its companion parts of the IEC 61158-5 subseries cancel and replace IEC 61158-5:2003. This edition of this part constitutes a technical revision.

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

a) deletion of the former Type 6 fieldbus for lack of market relevance;
b) addition of new types of fieldbuses;

c) partition of part 5 of the third edition into multiple parts numbered -5-2, -5-3, …

This bilingual version (2015-06) corresponds to the English version, published in 2007-12.

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

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

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

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

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

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

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

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

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

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 5-8: Application layer service definition – Type 8 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 viewed as a "window between corresponding application programs."

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 8 fieldbus. The term "time-critical" is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible service provided by the Type 8 fieldbus application layer in terms of

a) an abstract model for defining application resources (objects) capable of being manipulated by users via the use of the FAL service,

b) the primitive actions and events of the service;

c) the parameters associated with each primitive action and event, and the form which they take; and

d) the interrelationship between these actions and events, and their valid sequences.

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

1) the FAL user at the boundary between the user and the application layer of the fieldbus reference model, and

2) Systems Management at the boundary between the application layer and Systems Management of the fieldbus reference model.

This standard specifies the structure and services of the Type 8 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

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

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

1.2 Specifications

The principal objective of this 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. It is this latter objective which gives rise to the diversity of services standardized as the various Types of IEC 61158, and the corresponding protocols standardized in subparts of IEC 61158-6.

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

a) the sizes and octet ordering of various multi-octet service parameters, and

b) the correlation of paired request and confirm, or indication and response, primitives.

1.3 Conformance

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

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill the Type 8 application layer services as defined in this standard.

2 Normative references

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

IEC 60559, *Binary Floating-point Arithmetic for Microprocessor Systems*

ISO/IEC 646, *Information technology – ISO 7–bit coded character set for information interchange*

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

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

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

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

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

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

3 Terms and definitions

For the purposes of this document, the following terms, definitions, symbols, abbreviations and conventions apply.

3.1 ISO/IEC 7498-1 terms

This standard is partly based on the concepts developed in ISO/IEC 7498-1, and makes use of the following terms defined therein:

a) application entity
b) application process
c) application protocol data unit
d) application process invocation
e) application transaction
f) real open system
g) transfer syntax

3.2 ISO/IEC 8822 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8822 apply:

a) abstract syntax
b) presentation context

3.3 ISO/IEC 9545 terms

For the purposes of this document, the following terms as defined in ISO/IEC 9545 apply:

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

3.4 ISO/IEC 8824 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824 apply:

a) object identifier
b) type

3.5 Fieldbus data-link layer terms

For the purposes of this document, the following terms, as defined in IEC 61158-3-8 and IEC 61158-4-8, apply.

a) DLCEP
b) DLSAP
c) link
d) node

3.6 Fieldbus application layer specific definitions

3.6.1
access protection
limitation of the usage of an application object to one client

3.6.2
allocate
take a resource from a common area and assign that resource for the exclusive use of a specific entity

3.6.3
application
function or data structure for which data is consumed or produced

3.6.4
application layer interoperability
capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.6.5
application objects
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

3.6.6
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.6.7
application process identifier
distinguishes multiple application processes used in a device

3.6.8
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 Definition). 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 dynamically create and delete application objects and their corresponding definitions.

3.6.9
application process object class
a class of application process objects defined in terms of the set of their network-accessible attributes and services

3.6.10
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation. This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities

3.6.11
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.6.12
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.

3.6.13
attribute
description of an externally visible characteristic or feature of an object

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

3.6.14
behaviour
indication of how an object responds to particular eventss

3.6.15
class
a set of objects, all of which represent the same kind of system component

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

3.6.16
class attribute
attribute that is shared by all objects within the same class

3.6.17
class code
unique identifier assigned to each object class

3.6.18
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service

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

3.6.19
client
a) object which uses the services of another (server) object to perform a task

b) initiator of a message to which a server reacts

3.6.20
control commands
action invocations transferred from client to server to clear outputs, freeze inputs and/or synchronise outputs

3.6.21
conveyance path
unidirectional flow of APDUs across an application relationship

3.6.22
cyclic
repetitive in a regular manner

3.6.23
data consistency
means for coherent transmission and access of the input- or output-data object between and within client and server

3.6.24
dedicated AR
AR used directly by the FAL User

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

3.6.25
device
physical hardware connected to the link

NOTE A device may contain more than one node.

3.6.26
dynamic AR
AR that requires the use of the AR establishment procedures to place it into an established state

3.6.27
end node
producing or consuming node

3.6.28
endpoint
one of the communicating entities involved in a connection

3.6.29
engineering
abstract term that characterizes the client application or device responsible for configuring an automation system via interconnecting data items

3.6.30
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.6.31
error class
general grouping for related error definitions and corresponding error codes

3.6.32
error code
identification of a specific type of error within an error class

3.6.33
event
an instance of a change of conditions

3.6.34
group
(a) collection of objects

(b) address that identifies more than one entity

3.6.35
interface
a) shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

b) collection of FAL class attributes and services that represents a specific view on the FAL class

3.6.36
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.

3.6.37
index
address of an object within an application process

3.6.38
instance
the actual physical occurrence of an object within a class that identifies one of many objects within the same object class

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

3.6.39
instance attributes
attribute that is unique to an object instance and not shared by the object class

3.6.40
instantiated
object that has been created in a device

3.6.41
manufacturer ID
identification of each product manufacturer by a unique number

3.6.42
management information
network-accessible information that supports managing the operation of the fieldbus system, including the application layer

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

3.6.43
member
piece of an attribute that is structured as an element of an array

3.6.44
method
<object> a synonym for an operational service which is provided by the server ASE and invoked by a client

3.6.45
network
a set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.6.46
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 behaviour

3.6.47
object specific service
service unique to the object class which defines it

3.6.48
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.6.49
point-to-point connection
connection that exists between exactly two application objects

3.6.50
pre-defined AR endpoint
AR endpoint that is defined locally within a device without use of the create service

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

3.6.51
pre-established AR endpoint
AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.6.52
property
descriptive information about an object

3.6.53
provider
source of a data connection

3.6.54
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 the number of subscribers and it may publish its APDUs using a dedicated AR.

3.6.55
push publisher
type of publisher that publishes an object in an unconfirmed service request APDU

3.6.56
push subscriber
type of subscriber that recognizes received unconfirmed service request APDUs as published object data

3.6.57
resource
a processing or information capability of a subsystem

3.6.58
server
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

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

3.6.59
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.6.60
subscriber
role of an AREP in which it receives APDUs produced by a publisher

3.7 Abbreviations and symbols

AE	Application entity
AL	Application layer
ALME	Application layer management entity
ALP	Application layer protocol
APO	Application object
AP	Application process
APDU	Application protocol data unit
API	Application process identifier
AR	Application relationship
AREP	Application relationship endpoint
ASE	Application service element
CIM	Computer integrated manufacturing
Cnf	Confirmation
CR	Communication relationship
CREP	Communication relationship endpoint
DL-	(as a prefix) Data-link-
DLC	Data-link connection
DLCEP	Data-link connection endpoint
DLL	Data-link layer
DLM	Data-link-management
DLSAP	Data-link service access point
FAL	Fieldbus application layer
FIFO	First in first out
ID	Identifier
Ind	Indication
LME	Layer management entity
OSI	Open Systems Interconnect
PDU	Protocol data unit
Req	Request
Rsp	Response
SAP	Service access point

3.8 Conventions

3.8.1 Overview

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

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

3.8.2 General conventions

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

3.8.3 Conventions for class definitions

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

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

(1) The "FAL ASE:" entry is the name of the FAL ASE that provides the services for the class being specified.

(2) The "CLASS:" entry is the name of the class being specified. All objects defined using this template will be an instance of this class. The class may be specified by this standard, or by a user of this standard.

(3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the FAL ASE that will provide the services for this class. When qualified by the identity of its FAL ASE, it unambiguously identifies the class within the scope of the FAL. The value "NULL" indicates that the class cannot be instantiated. Class IDs between 1 and 255 are reserved by this standard to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2048 are allocated for identifying user defined classes.

(4) The "PARENT CLASS:" entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited for the class being defined, and therefore do not have to be redefined in the template for this class.

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

(5) The "ATTRIBUTES" label indicate that the following entries are attributes defined for the class.

a) Each of the attribute entries contains a line number in column 1, a mandatory (m) / optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label in column 3, a name or a conditional expression in column 4, and optionally a list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.

b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.

c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting is used to specify

i) fields of a structured attribute (4.1, 4.2, 4.3),

ii) attributes conditional on a constraint statement (5). Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2).

iii) the selection fields of a choice type attribute (6.1 and 6.2).

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

a) An (m) in column 2 indicates that the service is mandatory for the class, while an (o) indicates that it is optional. A (c) in this column indicates that the service is conditional. When all services defined for a class are defined as optional, at least one has to be selected when an instance of the class is defined.

b) The label "OpsService" designates an operational service (1).

c) The label "MgtService" designates an management service (2).

d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

3.8.4 Conventions for service definitions

3.8.4.1 General

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

3.8.4.2 Service parameters

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

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

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

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

M parameter is mandatory for the primitive

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

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

— (blank) parameter is never present.

S parameter is a selected item.

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

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

b) an indication that some note applies to the entry:
"(n)" indicates that the following note "n" contains additional information pertaining to the parameter and its use.

3.8.4.3 Service procedures

The procedures are defined in terms of

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

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

4 Concepts

4.1 Overview

This subclause describes 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.

4.2 Architectural relationships

4.2.1 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, its architectural relationship to the lower layers is different, as shown in Figure 1.

- The FAL includes 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 specifying the FAL.
- The FAL directly uses the services of the underlying layer. The underlying layer may be the data link layer 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.

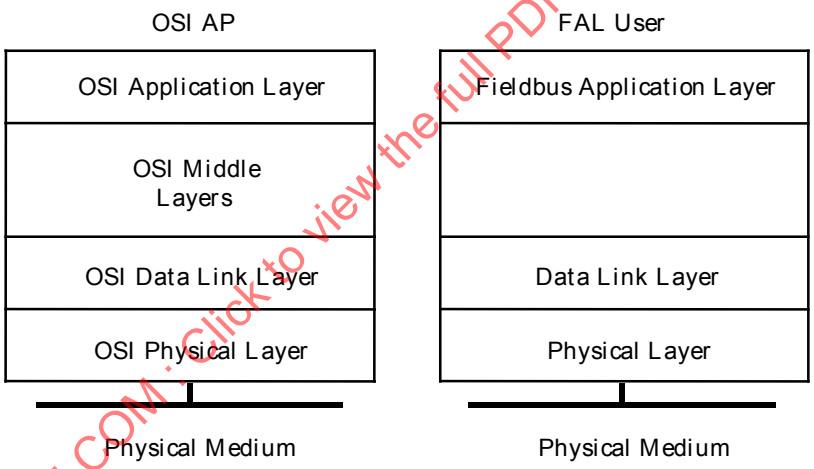


Figure 1 – Relationship to the OSI basic reference model

4.2.2 Relationships to other fieldbus entities

4.2.2.1 General

The Type 8 Fieldbus Application Layer (FAL) architectural relationships, as illustrated in Figure 2, have been designed to support the interoperability needs of time-critical systems distributed within the fieldbus environment.

Within this environment, the FAL provides communications services to time-critical and non-time-critical applications located in Type 8 devices.

In addition, the FAL directly uses the data-link layer to transfer its application layer protocol data units. It does this using a set of data transfer services and a set of supporting services used to control the operational aspects of the data-link layer.

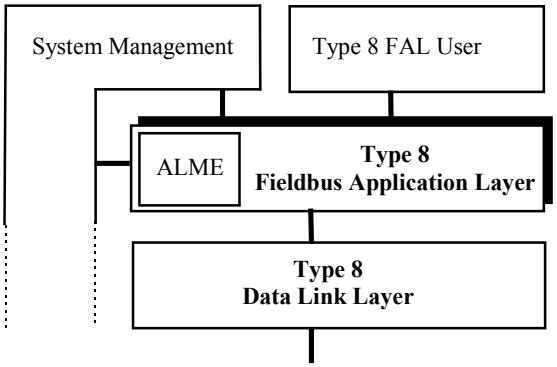


Figure 2 – Architectural positioning of the Type 8 FAL

4.2.3 Mapping of functional capabilities

Two communication mechanisms are supported:

- cyclic transmission of data in a high efficient manner using a publisher subscriber model,
- non cyclic transmission of data using a client server communication model

AREPs, which act as push publisher or push subscriber, are used for cyclic transmission.

AREPs, which act as client and server (peer), are used for non cyclic data transmission. A specific transfer syntax is used for their APDUs. The non cyclic data transmission mechanism is described by a subset of those ASEs and services specified in 6.1.

4.2.3.1 Use of the Type 8 data-link layer

The fieldbus Application Layer (FAL) provides network access to fieldbus APs. It interfaces directly to the fieldbus data-link layer for transfer of its APDUs.

The data-link layer provides various types of services to the FAL for the transfer of data between Data-link endpoints (e.g. DLSAPs, DLCEPs).

4.2.3.2 Support for Type 8 applications

TYPE 8 applications are represented 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 application entity (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 Application Process Objects (APO). 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 them are provided by APO-specific application service elements (ASEs) contained within the FAL. These ASEs are designed to support user, function block, and management applications.

4.2.3.3 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.

4.2.3.4 Access to FAL layer management entities

One layer management entity (LME) may be present in each FAL entity on the network. 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 fieldbus Application Layer Management Entity (FALME) provides the FAL portion of the SMIB. How the SMIB is implemented is beyond the scope of this standard.

4.3 Fieldbus Application Layer structure

4.3.1 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 TYPE 8 environment.

The FAL differs from the other layers of OSI in two principal aspects:

- OSI defines a single type of application layer communications channel, the association, 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.

- The FAL uses the DLL to transfer its PDUs and not the presentation layer. Therefore, there is no explicit presentation context available to the FAL. Between the same pair (or set) of data link service access points the FAL protocol may not be used concurrently with other application layer protocols.

4.3.2 Fundamental concepts

The operation of time critical real open systems is modeled 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 data-link layer (part 3 and part 4) may provide for the distribution of time to all devices. They also may 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 modeled 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. Part 6 specifies the protocols necessary to convey these object services between applications.

4.3.3 Application Processes

4.3.3.1 Definition of the AP

In the TYPE 8 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 Application Process (AP). A fieldbus AP is a variation of an Application

Process as defined in ISO OSI Reference Model (ISO/IEC 7498). Fieldbus APs may be unambiguously addressed by at least one individual data-link layer service access point address. Unambiguously addressed, in this context, means that no other AP may simultaneously be located by the same address. This definition does not prohibit an AP from being located by more than one individual or group data link service access point address.

4.3.3.2 Communication services

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 part 6. The AP behavior associated with the services is specified by the AP.

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

Unconfirmed services, in contrast, 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.

4.3.3.3 AP interactions

4.3.3.3.1 General

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, 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 interactions between APs are referred to as client/server and publisher/subscriber interactions.

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

4.3.3.3.2 Client/server interactions

Client/server interactions are characterized by a bi-directional 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.

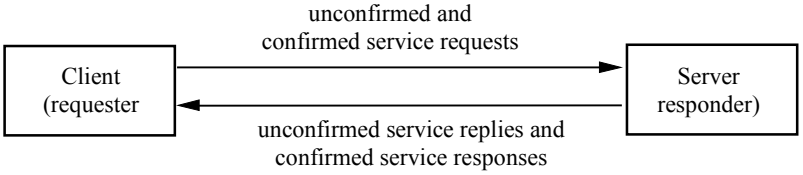


Figure 3 – Client/server interactions

4.3.3.3.3 Publisher/subscriber interactions

4.3.3.3.3.1 General

Publisher/subscriber interactions, on the other hand, involve a single publisher AP, and one subscriber APs. This type of interaction has been defined to support one model of interaction between APs, the "push" model. In the model, the setup of the publishing AP is performed by management and is outside the scope of this standard.

4.3.3.3.3.2 Push model interactions

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

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

Subscribers for the Push Model receive the published unconfirmed services distributed by publishers. Figure 4 illustrates the concept of the Push Model.

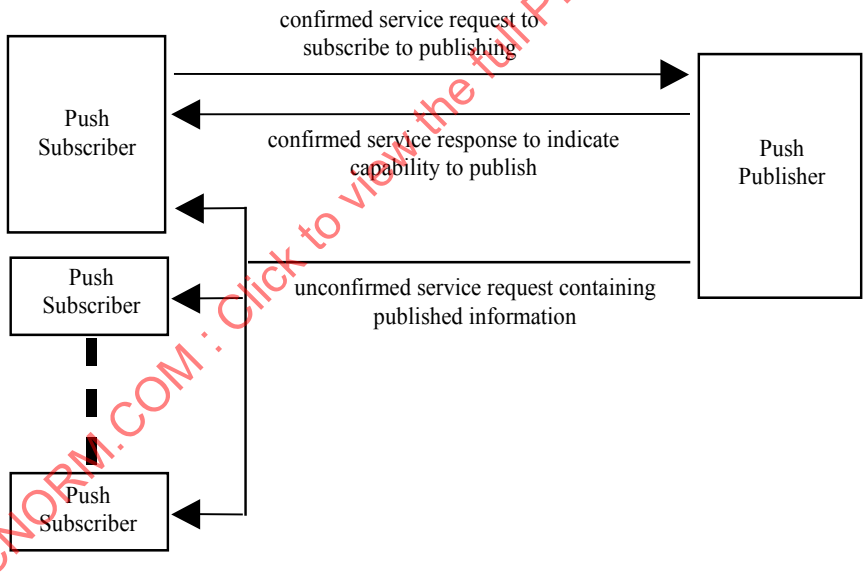


Figure 4 – Push model interactions

4.3.3.4 AP structure

The internals of APs may be represented by one, or more Application Process Objects (APOs) and accessed through one or more Application Entities (AEs). AEs provide the communication capabilities of the AP. For each 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.

4.3.3.5 AP class

An AP class is a definition of the attributes and services of an AP. The standard class definition for APs is defined in this standard. User defined classes also may be specified. Class identifiers (described in 3.8.2) are assigned from a set reserved for this purpose.

4.3.3.6 AP type

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

4.3.4 Application process objects

4.3.4.1 Definition of APO

An application process object (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 that are accessible through services of the FAL. APOs are used to represent these capabilities to other APs in a fieldbus system.

From the perspective of the FAL, an APO is modeled as a network accessible object contained within an AP or within another APO (APOs may contain other APOs). APOs provide the network definition for objects contained within an AP that are remotely accessible. The definition of an APO includes an identification of the FAL services that can be used by remote APs for remote access. The FAL services, as shown in Figure 5, are provided by the FAL communications entity of the AP, known as the FAL Applications Entity (FAL AE).

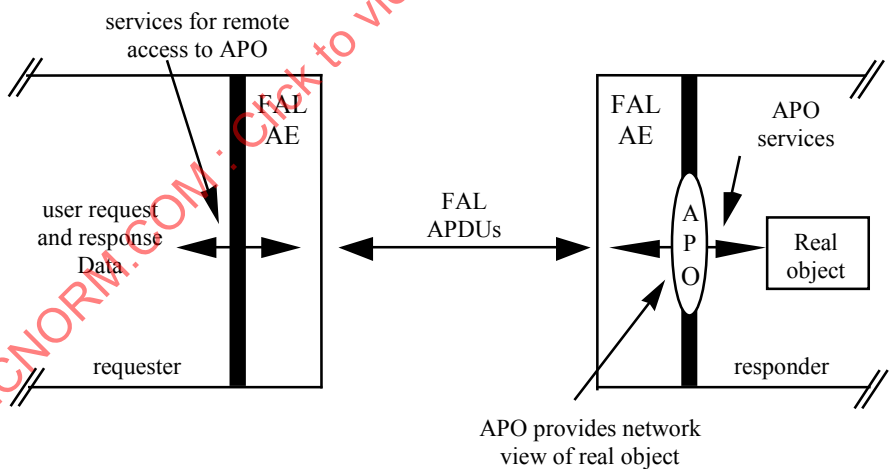


Figure 5 – APOs services conveyed by the FAL

In Figure 5, 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.

4.3.4.2 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 61158-6-8 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 3.8.2 may be used for this purpose. The method for registering or otherwise making these new class definitions available for public use is beyond the scope of this standard.

4.3.4.3 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

(1) selecting the attributes from its APO Class template that are to be accessible from the real object;

(2) assigning values to one or more attributes indicated as key in the template. Key attributes are used to identify the APO;

(3) assigning values to zero, one, or more non-key attributes for the APO. Non-key attributes are used to characterize the APO;

(4) selecting the services from the template that may be used by remote APs to access the real object.

Subclause 3.8.2 of this standard 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 that indicate whether or not the attribute is to be present in an APO.

4.3.4.4 APO types

APO types provide the mechanism for defining standard APOs.

As described above in the previous subclauses, 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 contain 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.

4.3.5 Application entities

4.3.5.1 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 Application Service Elements (ASE), such that the FAL services provided to an AP are defined by the ASEs its FAL AE contains. Figure 6 illustrates this concept.

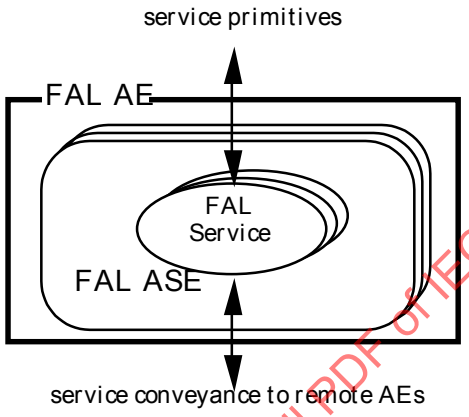


Figure 6 – Application entity structure

4.3.5.2 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.

4.3.6 Application service elements

4.3.6.1 General

An application service element (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.

4.3.6.2 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 may be used by the user to associate the response with the appropriate request.

NOTE This user-specified Invoke ID is distinct from any Instance ID that the requesting AP and its FAL AE may itself use to associate request with confirm, indication with response, or command APDU with reply APDU. See 1.2.

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.

4.3.6.3 Definition of FAL ASEs

4.3.6.3.1 General

A modular approach has been taken in the definition of FAL ASEs. The ASEs defined for the FAL are also object-oriented. In general, ASEs provide a set of services designed for one specific object class or for a related set of classes. Common object management ASEs, when present, provide a common set of management services applicable to all classes of objects.

To support remote access to the AP, the Application Relationship ASE is defined. It provides services to the AP for defining and establishing communication relationships with other APs, and it provides services to the other ASEs for conveying their service requests and responses.

Each FAL ASE defines a set of services, APDUs, and procedures that operate on the classes that it represents. Only a subset of the ASE services may be provided to meet the needs of an application. Profiles may be used to define such subsets. Definition of profiles is beyond the scope of this standard.

APDUs are sent and received between FAL ASEs that support the same services. Each FAL AE contains, at a minimum, the AR ASE and at least one other ASE. Figure 7 illustrates an example set of the FAL ASEs and their architectural relationships. All APO ASEs follow this example.

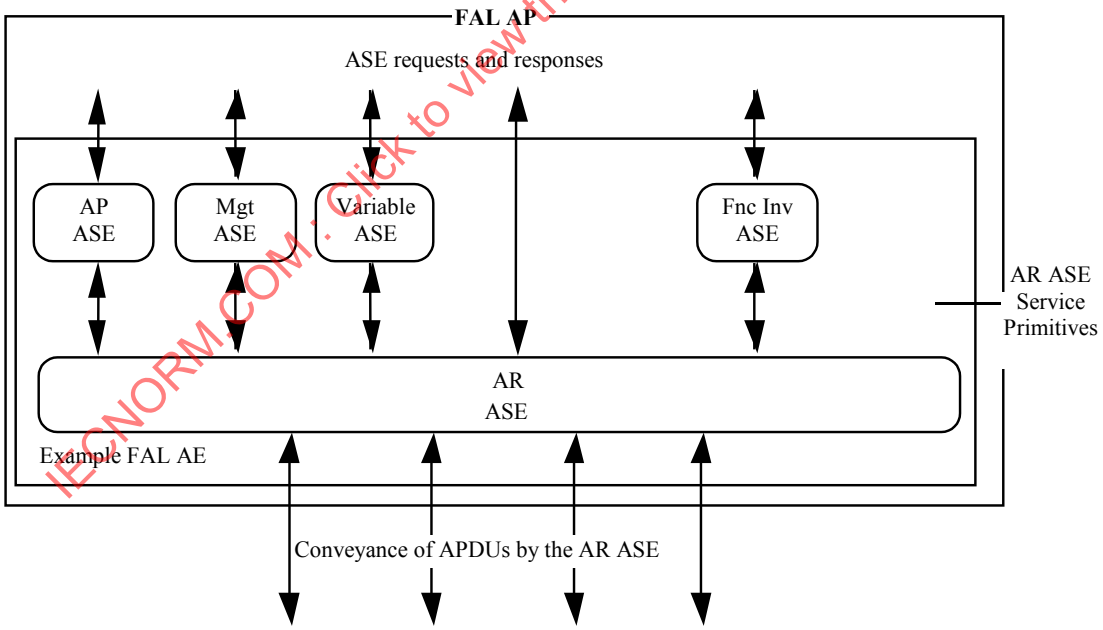


Figure 7 – Example FAL ASEs

4.3.6.3.2 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.

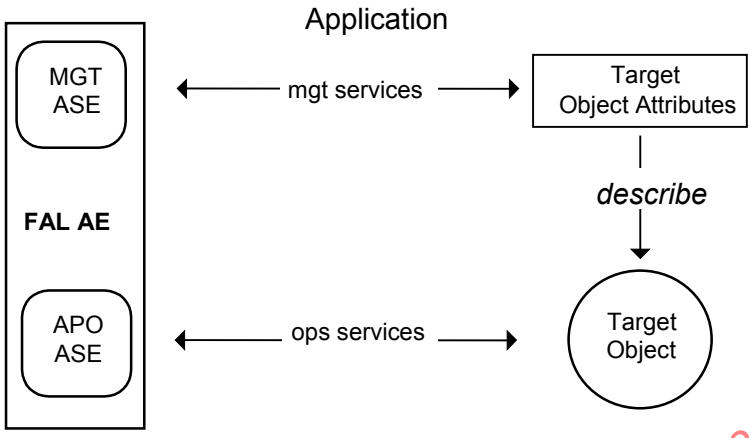


Figure 8 – FAL management of objects

4.3.6.3.3 AP ASE

An AP ASE may be specified for the identification and control of FAL APs. The attributes defined by the AP ASE specify characteristics of the AP about its manufacturer, and list its contents and capabilities.

4.3.6.3.4 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.

4.3.6.3.5 AR ASE

An AR ASE is specified to establish and maintain application relationships (ARs) that are used to convey FAL APDUs between/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, for conveying APDUs for the AE, and for indicating the local status of the AR to the user. In addition, local services may be defined for accessing certain aspects of AR endpoints.

4.3.6.4 FAL service conveyance

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

To accomplish the task of conveying service requests and responses, three types of activities for the sending user and three corresponding types for the receiving user are defined. At the sending user, they accept service requests and responses to be conveyed. Second, they select the type of FAL APDU that will be used to convey the request or response and encode the service parameters into its body portion. Then they submit the encoded APDU body to the AR ASE for conveyance.

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

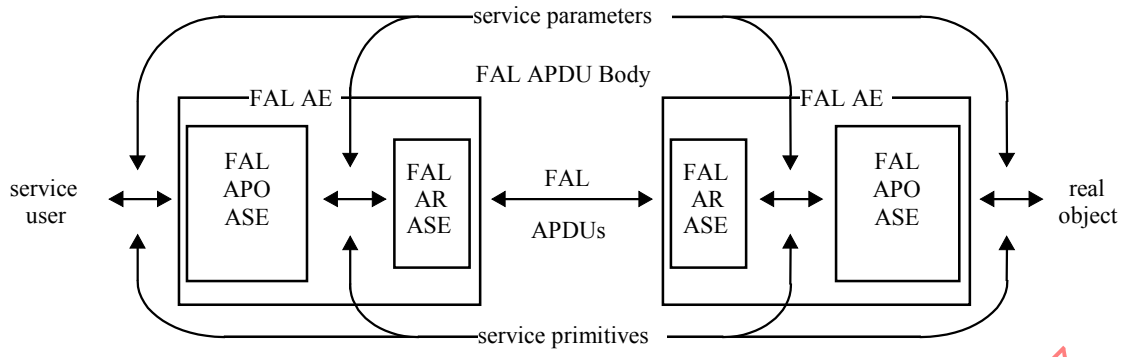


Figure 9 – ASE service conveyance

4.3.6.5 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 the presentation layer. Therefore, an alternate mechanism is provided for the FAL by each of the specific types of Communication Models.

4.3.7 Application relationships

4.3.7.1 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.

4.3.7.2 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.

4.3.7.3 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.

4.3.7.4 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-to-1. Accessing objects through ARs

ARs provide access to APs and the objects within them through 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 are selected from those defined for the AE.

4.3.7.5 AR conveyance paths

ARs are modeled as one or two conveyance paths between AR endpoints. Each conveyance path conveys APDUs in one direction between one or more AR endpoints. Each receiving AR

endpoint for a conveyance path receives all APDUs transmitting on the AR by the sending AR endpoint.

4.3.7.6 AREP roles

Because 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 only in one direction.

4.3.7.7 AREP buffers and queues

AREPs may be modeled 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 data-link layer 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 whether or not they were transmitted. Once an APDU is written into a buffer for transfer, it is preserved in the buffer until the next APDU to be transmitted replaces it. While in the buffer, an APDU may be read more than once without deleting it from the buffer or changing its contents.

At the receiving end, the operation is similar. 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 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.

4.3.7.8 User-triggered and scheduled conveyance

Another characteristic of an AREP is when they convey service requests and responses. AREPs that convey them upon submission by the user are called user-triggered. Their conveyance is asynchronous with respect to network operation.

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

4.3.7.9 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 dynamically established. Figure 10 depicts these two cases. AREPs also may require both dynamic definition and establishment or they may be dynamically defined such that they may be used without any establishment (they are defined in an established state).

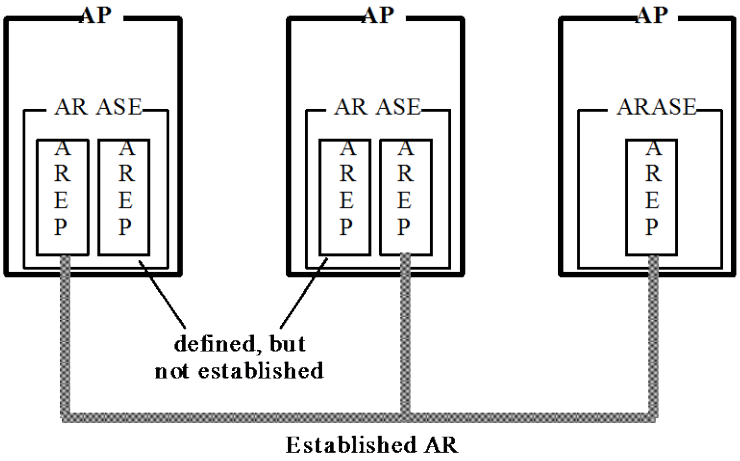


Figure 10 – Defined and established AREPs

4.3.7.10 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.

4.4 FAL naming and addressing

4.4.1 General

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

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

4.4.2 Identifying objects accessed through the FAL

4.4.2.1 General

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

Identifiers for APs and APOs within the FAL are defined as key attributes in the class definitions for APOs. Within these APO definitions, two types of key attributes are commonly used, names and numeric identifiers.

4.4.2.2 Names

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

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

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 not as easy to understand as descriptive names.

4.4.2.3 Numeric identifiers

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

4.4.3 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.

4.5 Architecture summary

The summary of the FAL architecture is presented in this Clause. Figure 11 illustrates the major components of the FAL architecture and how they relate to each other.

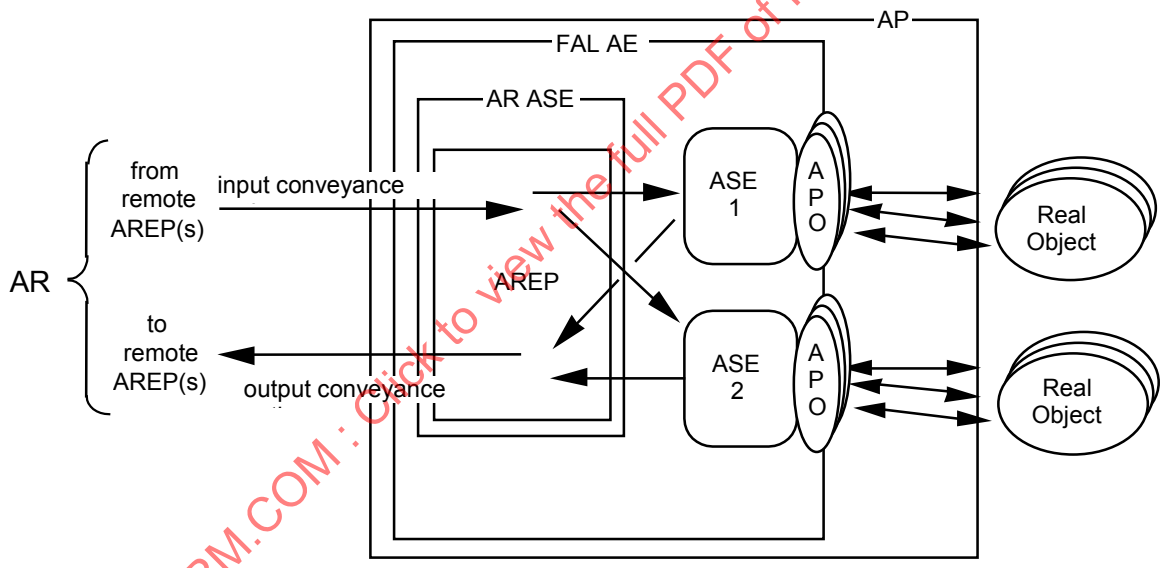


Figure 11 – FAL architectural components

Figure 11 depicts an AP that communicates through the FAL AE. The AP represents its internal real objects as APOs for remote access to them. Two ASEs are shown that provide the remote access services to their related APOs. 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.

4.6 FAL service procedures

4.6.1 FAL confirmed service procedures

4.7 Notional FAL service procedures [INFORMATIVE]

NOTE This subclause describes one way in which this abstract service definition might be realized in a concrete protocol specification and conforming implementations. See 1.2.

4.7.1 Notional FAL confirmed service procedures

The requesting AL-service user invokes a confirmed-service request primitive of its FAL. The appropriate FAL ASE creates a transaction state machine to control the invocation of the service, assigns an InstanceID and timeout time to that state machine, builds the related confirmed-service-request APDU body including that Instance ID, and conveys it on the specified AR.

Upon receipt of the confirmed-service-request APDU body, the receiving ASE decodes it. If a protocol error did not occur, the receiving ASE creates a transaction state machine to manage the expected response, assigns an independent (second) InstanceID to that state machine, then delivers a confirmed-service indication primitive to it's AL-service user, with the (second) InstanceID as an extra implementation parameter.

If the responding AL-service user is able to successfully process the request, the user returns a confirmed-service response (+) primitive, identifying the transaction by the InstanceID presented as part of the stimulating indication primitive.

If the responding user is unable to successfully process the request, the service fails and the user issues a confirmed-service response (–) primitive indicating the reason for failure, again identifying the transaction by the InstanceID presented as part of the stimulating indication primitive.

Whichever response the AL-service users chooses, the receiving ASE has available both the information from the response primitive and that from the associated indication primitive when it forms the APDU to be returned to the initiating ASE.

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, either of which contains the (first) InstanceID of the original requesting APDU, and conveys it on the specified AR.

Upon receipt of the response or error APDU body, the initiating ASE uses the (first) InstanceID contained in the response or error APDU to associate the APDU with the appropriate state machine and request. Once that association has been made, the initiating ASE has available both the information from the received APDU and that from the associated request primitive. It delivers a confirmed-service confirmation primitive to the requesting FAL ASE which specifies success or failure, reports the reason for failure if a failure occurred, and cancels the associated transaction state machine.

If the timer associated with the sate machine expires before the initiating ASE receives the returned response or error APDU, the AR ASE delivers a confirmed-service confirmation(–) primitive to the requesting FAL ASE and cancels the associated transaction state machine.

4.7.2 Notional FAL unconfirmed service procedures

The requesting user submits an unconfirmed service request primitive to 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. .

4.8 Common FAL attributes

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

ATTRIBUTES:

1 (o) Key Attribute: Numeric Identifier

2 (o) Key Attribute: Name

Numeric identifier

This 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 or name or both. This attribute is required for the Data type model.

Name

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

4.9 Common FAL service parameters

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

AREP

This parameter specifies sufficient information to locally identify 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 (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

This user-provided parameter identifies this invocation of the service. It is used to associate service requests with responses. Therefore, no two outstanding service invocations should be identified by the same Invoke ID value.

NOTE 2 The mechanism by which an implementation of these abstract services correlates request and confirm primitives at one AREP, related indication and response primitives at a second AREP, and the APDUs that link remote AEs to implement such services, is a protocol issue as stated in 1.2. Any service Instance ID that might be carried in those APDUs is logically unrelated to this user-provided Invoke ID.

FAL ASE/FAL class

This parameter specifies the FAL ASE (e.g. AP, AR, Variable, Data type, Event, Function Invocation, Load Region) and the FAL Class within the ASE (e.g. AREP, Variable List, Notifier, Action).

Numeric ID

This parameter is the numeric identifier of the object.

Error info

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

Error class

This parameter indicates the general class of error. Valid values are specified in the definition of Error Code parameter, below.

Error code

This parameter identifies the specific service error.

Additional code

This optional parameter identifies the error encountered when processing the request specific to the object being accessed. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive.

4.10 APDU size

APDU size is Communication Model dependent.

5 Data type ASE

5.1 General

5.1.1 Overview

Data types specify the machine independent syntax for application data conveyed by FAL services. The fieldbus application layer supports the definition and transfer of both basic and constructed data types. Encoding rules for the data types specified in this Clause are provided in IEC 61158-6-8.

Basic types are atomic types that cannot be decomposed into more elemental types. Constructed types are types composed of basic types and other constructed types. Their complexity and depth of nesting is not constrained by this standard.

Data types are defined as instances of the data type class, as shown in Figure 12. 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.

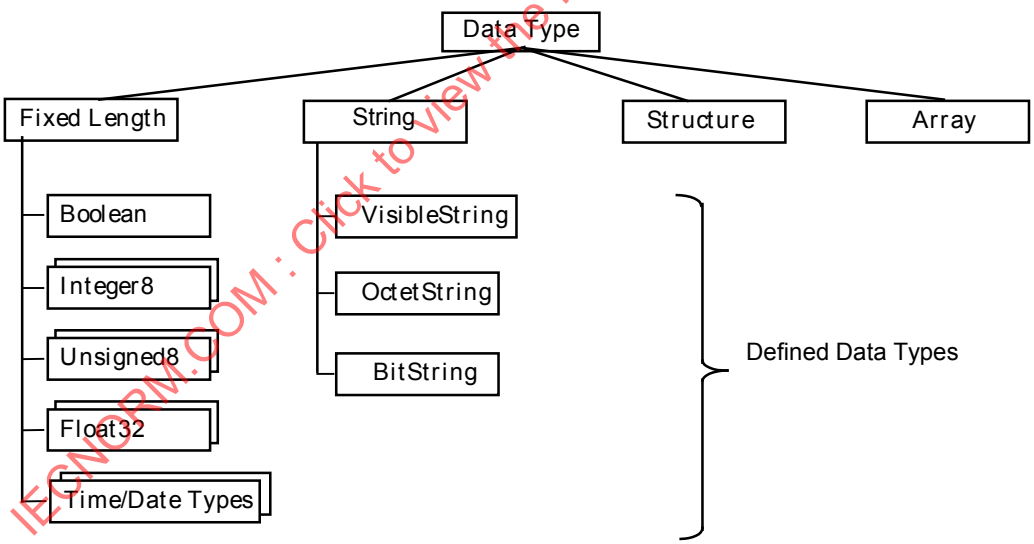


Figure 12 – Data type class hierarchy example

The data type definitions in Figure 12 are represented as a class/format/instance structure beginning with data type class entitled "Data type". The formats for data types are defined by the data type class and are represented in Figure 12.

The basic data classes are used to define fixed length and bitstring 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 and structures.

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 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 heterogenously 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

Only nesting level 1 is used.

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

The rules for encoding user data in FAL protocol data units is data type dependent. These rules are defined in IEC 61158-6-8. User-defined data types for which there are no encoding rules are transferred as a variable-length sequence 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)
PARENT CLASS:			TOP
ATTRIBUTES:			
1	(m)	Key Attribute:	Data type Numeric Identifier
2	(o)	Key Attribute:	Data type Name

5.2.1.2 Attributes

Data type Numeric Identifier

This attribute identifies the numeric identifier of the related data type.

Data type Name

This optional attribute identifies the name of the related data type.

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
4.1	Octet Length	=	1

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

5.3.1.2 Date types

5.3.1.2.1 BinaryDate

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	11
2	Data type Name	=	BinaryDate
3	Format	=	FIXED LENGTH

4.1	Octet Length	=	7

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned8 data type and gives the year.

5.3.1.2.2 BinaryDate2000

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	51
2	Data type Name	=	BinaryDate2000
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned16 data type and gives the year.

5.3.1.2.3 TimeOfDay

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
4	Format	=	FIXED LENGTH
4.1	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 and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date counting the days from January 1, 1984.

5.3.1.2.4 TimeDifference

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	13
2	Data type Name	=	TimeDifference
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4 or 6

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

5.3.1.3 Numeric types

5.3.1.3.1 Floating Point types – Float32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float32
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type has a length of four octets. The format for float32 is that defined by ISO 60559 as single precision.

5.3.1.3.2 Integer types

5.3.1.3.2.1 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.

5.3.1.3.2.2 Integer16

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	3
2	Data type Name	=	Integer16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

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

5.3.1.3.2.3 Integer32

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.3.3 Unsigned types

5.3.1.3.3.1 Unsigned8

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.3.3.2 Unsigned16

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	6
2	Data type Name	=	Unsigned16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

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

5.3.1.3.3.3 Unsigned32

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	7
2	Data type Name	=	Unsigned32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

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

5.3.2 String types

5.3.2.1 BitString

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	14
2	Data type Name	=	Bitstring
3	Format	=	STRING
5.1	Octet Length	=	1 to n

This string type is defined as a series of BitString8 elements.

5.3.2.2 OctetString

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

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet. The order of transmission is defined in IEC 61158-6-8.

5.3.2.3 VisibleString

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	9
2	Data type Name	=	VisibleString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

This type is defined as the ISO/IEC 646 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 Object management ASE

6.1.1.1 Overview

The FAL ASE Models each specify one or more related FAL APO classes as a collection of attributes and services. The FAL services defined for a class are used to manipulate, control, or access APOs of the class. The services which are supported by a specific APO are specified by the ListOfManagementServices attribute of the APO.

6.1.1.2 FAL management model specification

The management services specified by the FAL Management Model operate on FAL APOs and their attributes. FAL APO classes are defined within the FAL ASEs that are specified in the remaining clauses of this standard. Each FAL APO Class defined is described by a set of attributes and a set of services.

6.1.1.3 FAL management model services

6.1.1.3.1 Supported services

Management services are defined for managing APs and APOs. For simplicity APs and APOs are referred to as objects in the remainder of this subclause.

Get Attributes service is specified to indicate when its attributes can be read.

6.1.1.3.2 Get attributes service

####### 6.1.1.3.2.1 Service overview

This confirmed service is used to read the current values of a list of attributes for one or more objects of one or more classes defined for the AP. It operates in a "best effort" fashion, such that the service succeeds if the value for at least one requested attribute for one requested object is returned.

6.1.1.3.2.2 Service parameters

The service parameters for this service are shown in Table 1.

Table 1 – Get attributes service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
List of Objects and their Attributes	S	S (=)		
Key Attribute	M	M (=)		
List of Requested Attributes	M	M (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
More			M	M (=)
List of Responses			M	M (=)
List of Attribute Values			S	S (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

The argument contains the parameters of the service request.

List of objects and their attributes

This selection type parameter provides a list that identifies each object, its class, and the attributes that are being requested. Three values for identifying attributes are possible, all attributes, mandatory attributes only, or a list of individual attributes. It is present if the user is requesting attributes for a list of objects.

Key attribute

This parameter identifies the object for which attributes are being requested using one of the object's key attributes.

List of requested attributes

This parameter identifies one or more of the attributes for which values are being requested.

Result(+)

This selection type parameter indicates that the service request succeeded.

More

This Boolean parameter indicates, when FALSE, that responses for all requested attributes are being returned. If the value is TRUE, then only responses for a proper subset of the attributes are being returned. This situation will occur only when not all values will fit into a single response APDU. When it does occur, it is necessary for the user to submit an additional request for the missing attributes. Partial values for attributes are never returned.

List of responses

This parameter specifies a status indicator or a list of attribute values for each object specified in the request. The responses in the list are returned in the same order that the objects were specified in the request. The object associated with each response is only explicitly identified if the value of one of its key attributes was requested. If all responses could not be returned in a single APDU, the More parameter is set.

Error status

This parameter is returned if the get failed for all of the attributes of an object. It indicates the most probable reason why the operation failed using the error class and error code defined for the Get Attributes service.

List of attribute values

This parameter is returned if the Get succeeded for any one of the requested attributes. This parameter specifies a list of attribute values, each individually identified and complete, for the object requested.

Result(-)

This selection type parameter indicates that the service request failed.

6.1.1.3.2.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

The Get Attributes Service is a "best effort". Best effort means that the service succeeds if at least one value is returned. If the user is able to get one or more of the specified attribute values, and if they can be conveyed in a single APDU, they are returned in a Get Attributes Service response (+) primitive.

If they cannot be conveyed in a single APDU, the user returns those that can and sets the more flag in the Get Attributes Service response (+) primitive. The requester is then responsible for submitting additional requests identifying the remaining attributes to be retrieved.

If the user is unable to get the value for at least one attribute for one object in the list, the service fails and the user issues a Get Attributes Service response (-) primitive indicating the reason.

6.1.2 Application process ASE

6.1.2.1 Overview

This standard models an Application Process (AP). Application Processes represent the information and processing resources of a system that can be accessed through FAL services.

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

6.1.2.2 AP 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. The numeric identifier for the AP Object is always 0. The use of this reserved value permits management services to be used for AP objects without knowing their names.

FAL ASE:			**AP ASE**
CLASS:			**AP**
CLASS ID:			16
PARENT CLASS:			TOP
IDENTIFY, GET STATUS ATTRIBUTES:			
1	(m)	Attribute:	List Of AP Service Attributes
1.1	(m)	Attribute:	Manufacturer Identifier
1.1.1	(m)	Attribute:	Vendor Name
1.1.2	(m)	Attribute:	Model Identifier
1.1.3	(m)	Attribute:	Vendor Revision
1.2	(o)	Attribute:	AP Descriptor Reference
1.3	(m)	Attribute:	Network Access State
1.3.1	(m)	Attribute:	Service Access Status
1.3.2	(m)	Attribute:	Operational Status
SYSTEM MANAGEMENT ATTRIBUTES:			
2	(m)	Attribute:	List Of System Management Attributes
2.1	(m)	Attribute:	List Of AR Endpoint Info
2.1.1	(m)	Attribute:	Numeric Id
2.1.2	(m)	Attribute:	Configured Max FAL PDU Size Sending
2.1.3	(m)	Attribute:	Configured Max FAL PDU Size Receiving
2.1.4	(m)	Attribute:	Configured Max Outstanding Services Requesting
2.1.5	(m)	Attribute:	Configured Max Outstanding Services Responding
2.1.6	(m)	Attribute:	List of Supported Services
2.2	(m)	Attribute:	List Of In Use AR Endpoint Info
2.2.1	(m)	Attribute:	Context State
2.2.6	(m)	Attribute:	Outstanding Services Requesting Counter
2.2.7	(m)	Attribute:	Outstanding Services Responding Counter
OBJECT MANAGEMENT ATTRIBUTES:			
3	(m)	Attribute:	List Of Managed AP Attributes
3.1	(m)	Attribute:	List Of Supported APO Classes and Objects
3.1.1	(m)	Attribute:	List Header
3.1.1.1	(o)	Attribute:	Numeric Identifier = 0
3.1.1.2	(m)	Attribute:	ROM / RAM Flag (TRUE, FALSE)
3.1.1.3	(m)	Attribute:	Max Name Length
3.1.1.4	(m)	Attribute:	Access Protection Supported (TRUE, FALSE)
3.1.1.5	(m)	Attribute:	Version Of List
3.1.1.6	(m)	Attribute:	List Local Reference
3.1.2	(c)	Constraint	Data type Supported
3.1.2.1	(o)	Attribute:	Numeric Identifier = 1
3.1.2.2	(m)	Attribute:	Number Of Data type Objects
3.1.2.3	(m)	Attribute:	Local Reference
3.1.3	(c)	Constraint	Variable
3.1.3.1	(o)	Attribute:	Static Object Numeric Identifier
3.1.3.2	(m)	Attribute:	Number Of Static Objects (Variables)

3.1.3.3	(m)	Attribute:	Static Object Local Reference
3.1.5	(c)	Constraint	Function Invocation Supported
3.1.5.1	(o)	Attribute:	Function Invocation Numeric Identifier
3.1.5.2	(m)	Attribute:	Number Of Function Invocation Objects
3.1.5.3	(m)	Attribute:	Function Invocation Local Reference

SERVICES:

2	(o)	Ops Service:	Identify
3	(o)	Ops Service:	Get Status
5	(o)	Ops Service:	Initiate
6	(o)	Ops Service:	Terminate
8	(o)	Ops Service:	Reject

6.1.2.2.2 Identify, get status service attributes

List of AP service attributes

The following attributes are accessible using the Identify, Get Status Services. They are not otherwise accessible.

Manufacturer identifier

The Manufacturer Identifier is composed of the Vendor Name, Model Identifier, Vendor Revision, and optionally, the Serial Number of the AP. These attributes may be read using the Identify service.

Vendor name

This attribute specifies the name of the manufacturer of the AP.

Model identifier

This attribute specifies the model of the AP.

Vendor revision

This attribute specifies the revision level of the AP as defined by the manufacturer.

AP descriptor reference

This attribute is a reference to a generic description of the AP. The descriptor is an identifier for the characteristics of the AP independent of its use. Two APs, therefore, may share the same generic description. The value, and the permissible set of values for this attribute is user defined. This attribute is used in the Initiate service.

Network access state

The Network Access State attribute defines the ability of the AP to communicate. Two components are specified in this standard, Service Access Status and Operational Status. These attributes may be read using the Get Status service. The AP may choose to report them without being requested using the Status Notification service.

Service access status

This attribute specifies information about the state of the communication capabilities of the AP.

READY FOR COMMUNICATION All services may be used normally.

LIMITED NUMBER OF SERVICES Limited number of services may be supported in some cases (e.g. for small devices).

LOADING-NON-INTERACTING Object Definitions are in the process of being loaded locally. Therefore, the Begin Set Attributes service may not be used until the local load is complete and the state has been changed.

LOADING-INTERACTING Object Definitions are in the process of being loaded over the network (using the Set Attributes service). On the AR used for the loading, only the following services are used:

Abort	Get Attributes
Status	Set Attributes
Identify	End Set Attributes

OperationalStatus

This attribute gives the operational state of the real AP, as follows.

OPERATIONAL

PARTIALLY OPERATIONAL

NOT OPERATIONAL

NEEDS MAINTENANCE

6.1.2.2.3 System management attributes

List of system management attributes

The following attributes are accessible through System Management. They are used by all ASEs of the AP to enforce confirmed service flow control (see the state machines in IEC 61158-6-8). They are not otherwise accessible.

List of AR Endpoint Info

This attribute specifies the numeric identifiers and other configuration information (the AP Context attributes) of AREPs associated with the AP.

Numeric ID

This attribute specifies the numeric id for the AREP.

Configured max FAL PDU size sending

For non-segmenting ARs, this attribute specifies the configured maximum number of octets that can be sent from this AREP. Its value is equal to the maximum DLSDU size minus one that may be sent from this AREP. For segmenting ARs, it specifies the maximum number of 256-octet segments that can be sent on this AREP.

Configured max FAL PDU size receiving

For non-segmenting ARs, this attribute specifies the configured maximum number of octets that can be received on this AREP. Its value is equal to the maximum DLSDU size minus one that may be received on this AREP. For segmenting ARs, it specifies the maximum number of 256-octet segments that can be received on this AREP.

Configured max outstanding services requesting (ConfigMaxOsReq)

This attribute specifies the maximum number of outstanding confirmed services allowed in the client role of this AREP. Its value is configured before establishment of the AR. This attribute is used on client-server and peer-to-peer ARs (i.e. QUB) and corresponds to the number of sending transaction state machines.

Configured max outstanding services responding (ConfigMaxOsRsp)

This attribute specifies the maximum number of outstanding confirmed services allowed for this AREP as a server. Its value is configured before establishment of the AR. This attribute is used on client-server and peer-to-peer ARs (i.e., QUB) and corresponds to the number of receiving transaction state machines.

List of supported services

This attribute specifies the services that the FAL supports as a requester and as a responder. Support as a requester indicates that the FAL provides the ability for the AP to initiate confirmed and unconfirmed request primitives and to receive corresponding confirmed service confirmation primitives from the FAL. Support as a responder indicates that the FAL provides the ability for the AP to deliver confirmed and unconfirmed service indication primitives to the AP and receive confirmed service response primitives from the AP.

List of in-use AR endpoint info

This attribute specifies dynamic information for the AREPs associated with the AP which are participating in an established AR-I or which are involved in the AR establishment process.

Context state

This attribute specifies the state of the AREP as seen by the AP. The values are:

CLOSED

OPEN

OPENING-REQUESTING

OPENING-RESPONDING

Outstanding services requesting counter (OsReqCtr)

This attribute specifies the number of currently outstanding confirmed services on this AR as a requester.

Outstanding services responding counter (OsRspCtr)

This attribute specifies the number of currently outstanding confirmed services on this AR as a responder.

6.1.2.2.4 Object management attributes

List of managed AP attributes

This attribute specifies the AP attributes that are accessible using the services of the Management ASE.

List of supported APO classes and objects

This attribute specifies the Object Definitions maintained by the AP.

List header

This attribute describes the characteristics of the Object Definitions maintained by the AP.

Numeric ID

This attribute is the Numeric ID of the List Header. Its value is always 0.

ROM / RAM flag

This attribute, when TRUE, indicates that modifications to the Object Definitions are allowed.

Max name length

This attribute specifies the maximum length in octets for names of objects defined for this AP.

Access protection supported

This attribute, when TRUE, indicates that Access Protection is supported.

Version of list

This attribute indicates the current version of this list. Each update to the list of object definitions, either locally initiated or through the use of the SetAttributes service causes the version number to be incremented.

List local reference

This attribute is a reference to the list that has local significance, such as an address. The value FFFFFFFF hex indicates that no local reference is available.

XYZ object class supported constraint

This constraint selects object classes specified by XYZ. The attributes following the constraint apply to Object Definitions of the selected object class.

XYZ numeric ID

This attribute is the Numeric ID of the first object in a series of objects of the class(es) selected by the constraint. All objects of the selected class(es) are contained in the series. The first Numeric ID for Data types is always 1.

Number of XYZ entries

This attribute specifies the number of objects in the series.

XYZ local reference

This attribute is a reference to the beginning of object definitions that describe the objects in the series. The reference has local significance, such as an address. The value FFFFFFFF hex indicates that no local reference is available.

6.1.2.2.5 Services

Identify

This optional service is used to request manufacturer information from the AP.

Get Status

This optional service is used to request the network visible state from the AP.

Initiate

This optional service is used to open the AP context for an Application Relationship.

Terminate

This optional service is used to abruptly close the AP context for an Application Relationship.

Reject

This service is initiated internally by the AP ASE to indicate that a confirmed service request APDU has been received with a protocol error.

6.1.2.3 Application process 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:

Identify

Get Status

Initiate

Terminate

Reject

6.1.2.3.2 Identify service

6.1.2.3.2.1 Service overview

Information to identify an AP is read with this service.

NOTE The attribute ProfileNumber of the AP is transmitted with the Initiate service.

6.1.2.3.2.2 Service primitives

The service parameters for this service are shown in Table 2.

Table 2 – Identify

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Result(+)			S	S (=)
Invoke ID				U (=)
Vendor Name			M	M (=)
Model Identifier			M	M (=)
Vendor Revision			M	M (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Vendor name

This parameter specifies the value of the Vendor Name attribute of the AP.

Model identifier

This parameter specifies the value of the Model Identifier attribute of the AP.

Vendor revision

This parameter specifies the value of the Vendor Revision attribute of the AP.

Result(-)

This selection type parameter indicates that the service request failed.

6.1.2.3.2.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

6.1.2.3.3 Get status service

6.1.2.3.3.1 Service overview

The device/user state is read with the Get Status service.

6.1.2.3.3.2 Service primitives

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

Table 3 – Get status

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Result(+)			S	S (=)
Invoke ID				U (=)
Service Access Status			M	M (=)
Operational Status			M	M (=)
Local Detail			U	U (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service access status

This parameter specifies the value of the Service Access Status attribute of the AP.

Operational status

This parameter specifies the value of the Operational Status attribute of the AP.

Local detail

This parameter specifies the user defined state of the application.

Result(-)

This selection type parameter indicates that the service request failed.

6.1.2.3.3.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

6.1.2.3.4 Initiate service

6.1.2.3.4.1 Service overview

This service is used to establish a context between communicating APs for the exchange information on a single AR. The Initiate service negotiates the supported FAL services, the maximum outstanding services permitted, the maximum PDU length and the current version of the Object Definitions. The supported FAL services, the maximum outstanding services permitted, the maximum PDU length are AP system management attributes configured for the AREP.

6.1.2.3.4.2 Service primitives

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

Table 4 – Initiate

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Version Object Definitions Calling	M	M (=)		
AP Descriptor Calling	M	M (=)		
Access Protection Supported Calling	M	M (=)		
Password Calling	M	M (=)		
Access Groups Calling	M	M (=)		
Result(+)			S	S (=)
Version OD Called			M	M (=)
AP Descriptor Called			M	M (=)
Access Protection Supported Called			M	M (=)
Password Called			M	M (=)
Access Groups Called			M	M (=)
Result(-)			S	S (=)
Error Code			M	M (=)
Max PDU Length Sending Called				C
Max PDU Length Receiving Called				C
FAL Features Supported Called				C

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Version object definitions calling

This parameter specifies the version of the client's Object Definitions. Its value is null if the client does not contain Object Definitions.

AP descriptor calling

This parameter specifies the value of the calling AP's Descriptor Reference attribute if it has one. If it does not, its value is null.

Access protection supported calling

This parameter specifies the value of the calling AP's Access Protection Supported attribute if it has one. If it does not, its value is null.

Password calling

This parameter specifies the password to be used for the access to all objects of the server on this AR. If access with password is not to be used on this AR, its value is null.

Access groups calling

This parameter specifies the client's membership in specific access groups. The membership applies for the access to all objects of the server on this AR.

Result(+)

This selection type parameter indicates that the service request succeeded.

Version object definition called

This parameter specifies the version of the server's Object Definitions.

AP descriptor called

This parameter specifies the value of the called AP's Descriptor Reference attribute.

Access protection supported called

This parameter specifies the value of the called AP's Access Protection Supported attribute if it has one. If it does not, its value is null.

Password called

This parameter specifies the password to be used for the access to all objects of the client on this AR. If access with password is not to be used on this AR, its value is null.

Access groups called

This parameter specifies the server's membership in specific access groups. The membership applies for the access to all objects of the client on this AR.

Result(-)

This selection type parameter indicates that the service request failed.

Error code

This parameter provides the reason for the failure.

Reason	Meaning
Max FAL PDU Size Insufficient	The maximum PDU length is not sufficient for the communication.
Service Not Supported	The requested service or option is not supported by the server.
User Initiate Denied	The FAL user refuses to establish the connection.
Version Object Definition incompatible	The called and calling versions of the Object Definitions are not compatible.
Password Error	There is already an AR established with the same password, or the password is not valid.
AP Descriptor incompatible	The descriptor is not supported by the server.
Other	Reason other than any of those identified above.

Max PDU length sending called

This parameter specifies the maximum length of the FAL PDU to be sent that may be handled on this AR.

It is transmitted by the called FAL and is part of the Initiate.cnf primitive.

Max PDU length receiving called

This parameter specifies the maximum length of the FAL PDU to be received that may be handled on this AR.

If supported, it is transmitted by the called FAL and is part of the Initiate.cnf primitive.

FAL services supported called

This parameter identifies the optional AL services and the options supported by the server (see AR List).

If supported, it is transmitted by the called FAL and is part of the Initiate.cnf primitive.

6.1.2.3.4.3 Service procedure

This service operates through a queue.

The requesting user submits the service request primitive to its FAL AE and starts an associated timer to monitor the request. The AP ASE builds the Initiate Request APDU Body and conveys it on the specified AR using the AR Establish Service request primitive. It also creates a transaction state machine.

Upon receipt of the Initiate Request APDU Body in an AR Establish service indication primitive, the responding AP ASE decodes it. If a protocol error was encountered while decoding the received APDU Body, the ASE uses the AR Abort Service to abort the AR. If a protocol error did not occur, the ASE delivers an Initiate service indication primitive to its user.

If the responding user is able to successfully process the request, the user returns an Initiate service response (+) primitive.

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

The responding AP ASE builds an Initiate service response APDU Body for a response (+) primitive or an Initiate service error APDU Body for a response (-) primitive and conveys it on the specified AR using an AR Establish service response primitive.

Upon receipt of the returned APDU Body in an AR Establish service confirmation (+ or -) primitive the initiating ASE delivers an Initiate service confirmation primitive to its user which specifies success or failure, and the reason for failure if a failure occurred. It also cancels the corresponding transaction state machine.

However, if the AP's transaction timer expires before the corresponding response or error APDU is received, the AP may take corrective actions, which may include instructing its local ASE to cancel the transaction state machine. Such instruction to the ASE would occur through a locally defined interface.

6.1.2.3.5 Terminate service

6.1.2.3.5.1 Service overview

This service is used to release an existing AR between APs, or to terminate a subscriber/receiver AREP's involvement in an AR.

6.1.2.3.5.2 Service primitives

The service parameters for this service are shown in Table 5.

Table 5 – Terminate

Parameter name	Req	Ind
Argument		
AREP	M	M
Locally Generated		M
Terminate Identifier	M	M (=)
Reason Code	M	M (=)
Terminate Detail	U	U (=)

Argument

This parameter carries the parameters of the service invocation.

Locally generated

This parameter indicates whether the termination was generated locally or by the communication partner.

The value false is not allowed if the Terminate Identifier parameter has the value FAL and the Reason Code parameter has the value AR Error.

Terminate identifier

This parameter indicates where the reason for the termination has been detected. Possible values are:

FAL USER
FAL APO ASE
FAL AR ASE
DLL

Reason code

This parameter specifies the reason for the termination.

If the Terminate Identifier parameter has the value USER, the following values apply.

	Meaning
Disconnection	The connection is released by the FAL user.
Version Object Definition incompatible	The called and calling versions of the Object Definitions are not compatible.
Password Error	There is already an AR established with the same password, or the password is not valid.
Descriptor incompatible	The server's descriptor is not supported by the client.
Limited Services Permitted	The AP is in the Logical Status LIMITED-SERVICES-PERMITTED.

If the Abort Identifier parameter has the value FAL, the following values apply:

Reason	**Meaning**
AR Error	Faulty AR
User Error	Improper, unknown or faulty service primitive received from the FAL user
FAL PDU Error	Unknown or faulty FAL PDU received from the AR ASE
Connection State Conflict AR ASE	Improper AR ASE service primitive
AR ASE Error	Unknown or faulty AR ASE service primitive
PDU Size	PDU length exceeds maximum PDU length
Feature Not Supported	SERVICE-REQ PDU received from the AR ASE and service or option not supported as a server (see FAL Features Supported attribute in the AR)
Invoke ID Error Response	Confirmed service.rsp received from the FAL user and Invoke ID does not exist or CONFIRMED_SERVICE-RSP_PDU received from the AR ASE and Invoke ID does not exist
Max Services Overflow	CONFIRMED_SERVICE-REQ_PDU received from the AR ASE and Outstanding Services Counter Receiving ≥ Max Outstanding Services Receiving
Connection State Conflict	INITIATE-REQ_PDU received from the AR ASE
Service Error	The service in the response does not match the service in the indication or the service in the confirmation does not match the service in the request.
Invoke ID Error Request	CONFIRMED_SERVICE-REQ_PDU received from the AR ASE and Invoke ID already exists.

If the Terminate Identifier parameter has the value AR ASE or DLL, the Reason Code parameter is supplied by the AR ASE.

Terminate detail

This optional parameter specifies additional information about the abort reason (max. 16 octets). In case of error reports from the application, the meaning is defined in the profile.

6.1.2.3.5.3 Service procedure

The Unconfirmed Service Procedure specified in 4.6 applies to this service.

6.1.2.3.6 Reject service

6.1.2.3.6.1 Service overview

This service is used to inform the remote endpoint that a protocol error has been detected. It is generated by the AP ASE if an APDU has been received with an invalid service type code. When another APO ASE detects a protocol error, it internally requests the AP ASE to generate a Reject PDU.

6.1.2.3.6.2 Service primitives

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

Table 6 – Reject

Parameter name	Req	Ind
Argument		
AREP	M	M
Reject Code	M	M (=)
Original FAL Service Type	M	M (=)
Original Invoke ID	M	M (=)
Original PDU Body	U	U (=)

Argument

The argument contains the parameters of the service request.

Reject code

This parameter indicates the reason for rejection. The values of this may indicate:

Unsupported Service

AR Not Established

AR Not Defined

Max Outstanding Requests Exceeded

Max PDU Size Exceeded

Duplicate Invoke ID

Original FAL service type

This parameter specifies the service type of the rejected service request.

Original invoke ID

This parameter specifies the invoke id of the rejected service request.

Original PDU body

This optional parameter specifies some or all of the data of the APDU that was rejected. The amount of data included is determined by the user.

6.1.2.3.6.3 Service procedure

The Reject service is a service that operates through a queue or buffer. It may be initiated only by the FAL AP ASE.

If any APO ASE receives a confirmed request APDU that contains a protocol error, the AP ASE builds a Reject Request APDU and returns it on the specified AR.

Upon receipt of the Reject Request APDU, the receiving AP ASE delivers an AR-Reject.indication primitive through the FAL ASE that issued the confirmed send request primitive, and it issues a negative confirmation to the requesting user.

6.1.3 Application relationship ASE

6.1.3.1 Overview

6.1.3.1.1 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 application relationships (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 Application Service Element (ASE) that represents the class of the APO being accessed. Figure 13 illustrates this concept.

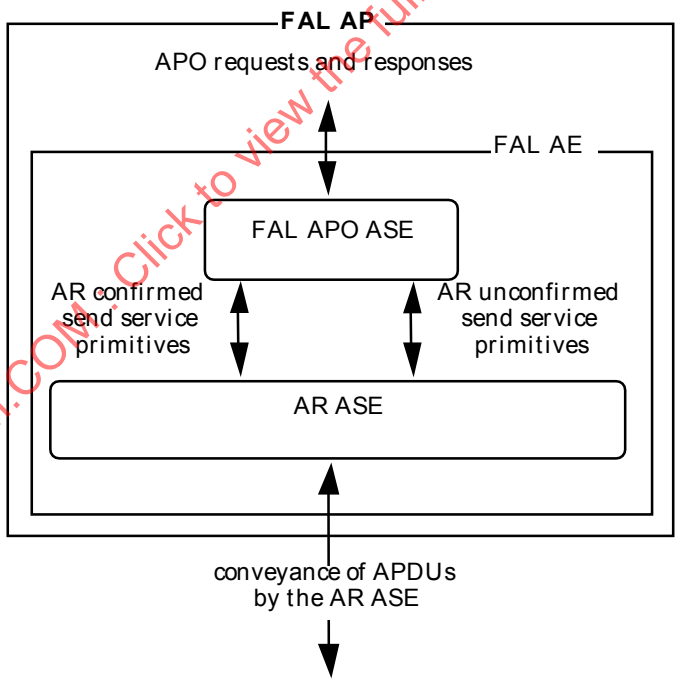


Figure 13 – The AR ASE conveys APDUs between APs

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.

6.1.3.1.2 Endpoint context

6.1.3.1.2.1 General

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, when combined with the definitions of the other endpoints defines an AR. To ensure communications compatibility among or between endpoints, each endpoint definition contains a set of compatibility-related characteristics. These characteristics need to 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, when combined with those used to specify compatibility, define the context of the endpoint. The endpoint context is used by the AR ASE to manage the operation of the endpoint and the conveyance of APDUs. The characteristics which comprise the endpoint context are described next.

6.1.3.1.2.2 Endpoint role

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

Table 7 and Table 8 summarize the characteristics and combinations of each of the AREP roles.

Table 7 – Conveyance of service primitives by AREP role

	Client	Server	Peer	Push Publisher	Push Subscriber
Send Service Req	X		X	X	
Recv Service Req		X	X		X
Send Service Rsp		X	X		
Recv Service Rsp	X		X		

Table 8 – Valid combinations of AREP roles involved in an AR

AREP Roles by Interaction Model	Client	Server	Peer	Push Publisher	Push Subscriber
Client/server					
Client		Yes	Yes		
Server			Yes		
Peer			Yes		
Publisher/subscriber					
Push Publisher					Yes
Push Subscriber					

6.1.3.1.2.3 Dedicated AR endpoints

AR endpoints that are dedicated provide their services directly to the FAL User. Although their behavior is the same as non-dedicated endpoints, the APDUs they convey contain no service specific protocol control information other than the AR control field.

FAL Users configured for dedicated ARs construct and transfer their data without using the services of the other FAL ASEs. The format and contents of the user data conveyed over dedicated ARs are known only through configuration.

6.1.3.1.2.4 Cardinality

The cardinality of an AR specifies, from the point of view of a client or publisher endpoint, 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-to-1. Clients are never capable of issuing a request and waiting for responses from multiple servers.

6.1.3.1.2.5 Conveyance model

6.1.3.1.2.5.1 General

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

conveyance paths

trigger policy, and

conveyance policy.

6.1.3.1.2.5.2 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.

To support the role of the application process, the endpoint is configured with either one or two conveyance paths. Endpoints that only send, or only receive, are configured with either a send or receive conveyance path, and those that do both are configured with both. ARs with a single conveyance path are called uni-directional, and those with two conveyance paths are called bi-directional.

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

6.1.3.1.2.5.3 Trigger policy

Trigger policy indicates when APDUs are transmitted by the data-link layer over the network.

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

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

This network scheduling mechanism can be either cyclic or one-time.

Network-scheduled ARs may also convey requests and responses asynchronously if the underlying layer has available bandwidth through use of the compel service. These transfers occur in addition to the scheduled transfers, and without being triggered from the network schedule.

6.1.3.1.2.5.4 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 replace its contents with new data. In buffered ARs, unconveyed or undelivered data that has been replaced is lost. Additionally, data contained in a buffer may be read multiple times without destroying its contents.

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, a new message will not be placed into the queue.

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 data-link layer. Or, 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 it is received, or by indicating its receipt and allowing the user to retrieve it in a separate operation. Another option is to require the user to detect that the buffer or queue has been updated.

6.1.3.1.3 Underlying communications services

6.1.3.1.3.1 General

The AR ASE conveys FAL APDUs using the capabilities of the underlying data-link layer. Several characteristics are used to describe these capabilities. This subclause provides a description of each. These characteristics are specific to the data-link mapping defined in IEC 61158-6-8. Their precise specification can be found there.

6.1.3.1.3.2 Connection-oriented services

The underlying layer may support AR endpoints by providing connection-oriented services. Endpoints configured to operate over connection-oriented services may be capable of opening and closing connections if the connections they use are not pre-configured to be open.

6.1.3.1.3.3 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 endpoints.

6.1.3.1.3.4 Cyclic and acyclic transfers

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

The underlying layer may also provide for the acyclic transfer of APDUs concurrently with cyclic transfer of APDUs between the same endpoints.

6.1.3.1.4 AR establishment

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

AR establishment can occur in one of two ways. First, ARs can be pre-established. Pre-established means that the AE that maintains the endpoint context is created when the AP is connected to the network. In this case, communications among the applications involved in the AR may take place without first having to explicitly establish the AR. Any AR may be defined to be pre-established.

Second, ARs can be pre-defined, but not pre-established. Pre-defined means that the characteristics of the AR are known to each endpoint, but that their contexts have not been synchronized for operation. In this case, the use of the FAL Establish service is required to synchronize them and open them for data transfer.

6.1.3.1.5 Application relationship classes

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

6.1.3.2 Application relationship endpoint class specifications

6.1.3.2.1 Formal model

The AR endpoint formal 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 subclause of this standard.

All AR endpoint attributes are accessible through system Management and not through the Object Management ASE services defined in this standard.

FAL ASE:			**AR ASE**
CLASS:			**AR ENDPOINT**
CLASS ID:			32
PARENT CLASS:			TOP
SYSTEM MANAGEMENT ATTRIBUTES:			
1	(m)	Attribute:	Local AP
2	(m)	Attribute:	FAL Revision
3	(m)	Attribute:	Dedicated (TRUE, FALSE)
4	(o)	Attribute:	Transfer Syntax
SERVICES:			
1	(o)	OpsService:	AR-Unconfirmed Send
3	(o)	OpsService:	AR-Establish
5	(o)	OpsService:	AR-Abort

6.1.3.2.2 System management attributes

Local AP

This attribute identifies the AP attached or configured to use the AREP using a local reference.

FAL revision

This 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

The attribute specifies, when TRUE, that the endpoint is dedicated. When TRUE, the services of the AR ASE are accessed directly by the FAL User.

Transfer syntax

This 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.

6.1.3.2.3 Services

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

AR-unconfirmed send

This optional service is used to send an unconfirmed service.

AR-establish

This optional service is used to establish (open) an AR.

AR-abort

This optional service is used to abort (abruptly terminate) an AR.

6.1.3.3 Application relationship ASE service specifications

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

AR-Unconfirmed Send

AR-Establish

AR-Abort

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

6.1.3.3.2 AR-unconfirmed send service

6.1.3.3.2.1 Service overview

This service is used to send AR-Unconfirmed request APDUs for FAL APO ASEs. The AR-Unconfirmed Send service may be requested at either endpoint of a one-to-one bi-directional AR, at the server endpoint of a one-to-one uni-directional AR.

6.1.3.3.2.2 Service primitives

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

Table 9 – AR-unconfirmed send

Parameter name	Req	Ind
Argument		
AREP	M	M
FAL Service Type	M	M (=)
FAL APDU Body	M	M (=)

Argument

The argument contains the parameters of the service request.

FAL service type

This parameter specifies the type of the service being conveyed.

FAL APDU body

This parameter specifies the service dependent body for the APDU.

6.1.3.3.2.3 Service procedure

The AR-Unconfirmed Send Service is a service that operates through a queue or buffer.

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.

If the AREP is queued, the AR ASE queues the APDU for submission to the lower layer. If the AR is buffered, the AR ASE replaces the previous contents of the buffer with the APDU contained in the service primitive.

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 DL 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 data-link layer.

NOTE The transmission schedule is managed by the underlying layer, not the AR-ASE. Refer to IEC 61158-3-8 and IEC 61158-4-8 for further details.

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.

6.1.3.3.3 AR-establish service

6.1.3.3.3.1 Service overview

This confirmed service operates in a pair-wise manner between two AR endpoints to synchronize their contexts and activate them for the transfer of APDUs.

The endpoint context may be created during establishment or prior to establishment through System Management.

6.1.3.3.3.2 Service primitives

The service parameters for this service are shown in Table 10.

Table 10 – AR-establish service

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
User Data	U	U (=)		
Result				
User Data			U	U (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

The argument contains the parameters of the service request.

AREP

This parameter specifies sufficient information to locally identify the AREP to be established.

User data

This optional parameter specifies user supplied data that is to be conveyed with the service request.

Result

This parameter indicates that the service request succeeded or failed.

AREP

This parameter specifies sufficient information to locally identify the AREP to be established.

User data

This optional parameter specifies user supplied data that is to be conveyed with the service response.

6.1.3.3.3.3 Service procedure

6.1.3.3.3.3.1 1-to-1 AR establishment

When used in support of 1-to-1 AREPs, the AR-Establish service causes Establish PDUs to be exchanged between calling and called DLSAPs as shown in Figure 14.

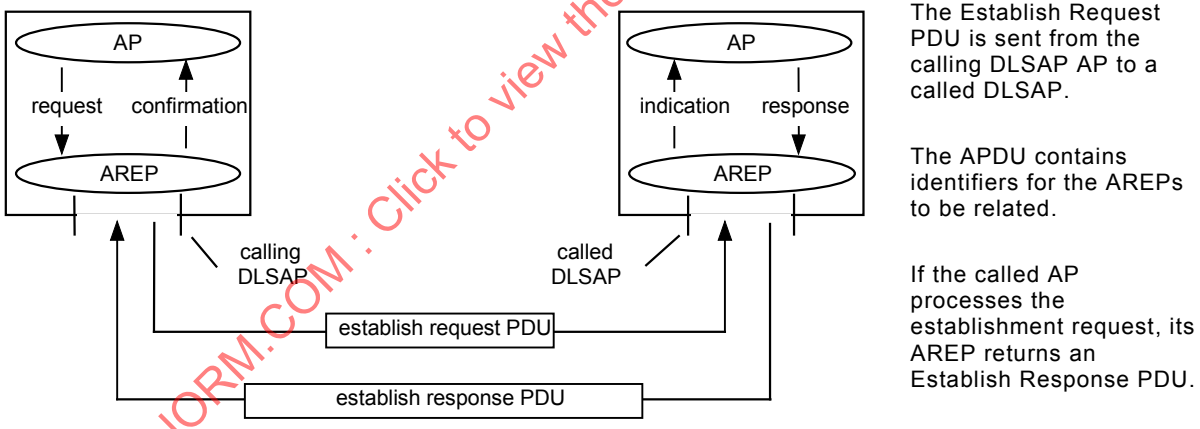


Figure 14 – 1-to-1 AR establishment

Upon receipt of an AR-Establish request service primitive, the calling FAL issues an Establish Request APDU to the called AP ASE that handles establishment of an AR. Upon receipt of an Establish indication primitive, the called AP ASE examines the parameters specified in it and returns the appropriate response in the AR-Establish response primitive.

6.1.3.3.3.2 Compatible AREP classes

Table 11 provides possible combinations of the AREP classes which may be related with the AR-Establish service. In this table the SERVER column contains a "-" to indicate that Server AREPs do not initiate AR establishment.

Table 11 – Valid combinations of AREP classes to be related

		Calling AREP Role		
		PEER	**CLIENT**	**SERVER**
Called AREP Role	PEER	YES	YES	--
	CLIENT	NO	NO	--
	SERVER	YES	YES	--

If the called AP decides to establish an AR, it issues an AR-Establish response primitive. The FAL returns the AR parameter which is used to refer to the AR being formed from the called service user. If the AR being established uses the Connection Oriented data-link layer and its explicit establishment is required, it is established before an Establish Response APDU is returned. The called AP then returns an Establish Response APDU with the Result(+) parameter to the calling AR_ASE. The calling AR_ASE issues an AR-Establish confirmation primitive in which the AR parameter is specified.

6.1.3.3.3.3 Conflict resolution

The normal establishment of ARs follows the general time-sequence for confirmed services. In the cases where each endpoint of an AR concurrently issues an AR-Establish request, a conflict arises. The algorithms for resolving conflicts of this type are specified in detail in IEC 61158-6-8.

6.1.3.3.4 AR-abort service

6.1.3.3.4.1 Service overview

The service is used by the FAL User to abruptly terminate an AR. It is always successful; the receiver of an Abort request or Abort request APDU always aborts the AR. This service may be used on any open AREP, whether or not the AR was pre-established or dynamically established using the establish service.

The AR-Abort Service is used to instruct the AR ASE to abruptly terminate all activity on an AREP and place it in the Closed state. Receipt of an AR-Abort request primitive causes the AR ASE to immediately close the AREP context and issue an Abort Request APDU to the remote AREPs. Receipt of an Abort Request APDU causes the AR ASE to immediately close the AR AREP context and deliver an AR-Abort request indication primitive to the user. The immediate close of each endpoint context causes all outstanding service requests to be cleared. All subsequent service primitives and APDUs received by the AR ASE for the aborted AR are discarded except for those of the AR-Establish service.

The AR-Abort service may be requested at either AREP of a one-to-one relationship.

The AR ASE may also initiate the abort service when it detects unrecoverable communication failures. In this case, the AR ASE delivers an AR-Abort indication primitive informing the user of the failure and closes the endpoint context.

6.1.3.3.4.2 Service primitives

The service parameters for this service are shown in Table 12.

Table 12 – AR-abort

Parameter name	Req	Ind
Argument		
AREP	M	M
Locally Generated		M
Originator	M	M (=)
Reason Code	M	M (=)
Additional Detail	U	U (=)

Argument

This parameter carries the information associated with the AR-Abort service.

Locally generated

This parameter specifies if the abort was locally generated, or not.

Originator

This parameter identifies the originator for the abort. Its valid values are DLL, FAL, or FAL-USER. The value DLL cannot be used in the request primitive.

Reason code

This parameter indicates the reason for the Abort. It may be supplied by either the provider or the user. One reason is defined: AR ASE error. Other reason code values may be supplied by the data-link layer, or by the user.

Additional detail

This optional parameter specifies user data that accompanies the indication. When used, the value submitted in the request primitive is delivered unchanged in the indication primitive.

6.1.3.3.4.3 Service procedure

The abort service is a service that operates through a queue or buffer.

If the user wishes to abort an AR, it submits an abort request primitive to its FAL AR ASE. If an AR ASE detects an unrecoverable local failure or a communication failure, it delivers an abort indication primitive to the endpoint user.

The FAL AR ASE builds an abort request APDU and conveys it on the specified AR if it is capable of doing so. It also transitions the AREP state to CLOSED

Upon receipt of the Abort Request APDU, each remote FAL AR ASE transitions its AREP to CLOSED and delivers an abort indication primitive to its user.

6.1.3.3.5 AR-data-send-acknowledge service

This service is defined for the FAL to provide the FAL user a service for transparent data transfer. This service can be used to exchange arbitrary data over a DLL.

For this service no specific ARPM state machine is available. The data flow is controlled by the FAL user.

Table 13 – AR-Data-Send-Acknowledge service parameters

Parameter name	Req	Ind	Cnf
Argument			
Destination Address	M	M (=)	
Source Address	M	M (=)	
Destination Node	M	M (=)	
Source Node	M	M (=)	
Destination Subnode	M	M (=)	
Source Subnode	M	M (=)	
Protocol Code	M	M (=)	
Block Number	M	M (=)	
Data	M	M (=)	
Result(+)			S
Result(-)			S
Error Info			M

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

The argument contains the parameters of the service request.

Destination address

This parameter specifies the logical destination address of the requested device. The broadcast address is FF FF FF.

Source address

This parameter specifies the logical source address of the local device.

Destination node

This parameter specifies the destination node of the requested device.

Source node

This parameter specifies the destination node of the local device.

Destination subnode

This parameter specifies the destination subnode of the requested device.

Source subnode

This parameter specifies the destination subnode of the local device.

Protocol code

This parameter specifies the FAL user protocol.

Block number

This parameter is the number of the PDU segment.

Data

This parameter specifies the PDU that has to be exchanged.

6.1.4 Variable ASE

6.1.4.1 Overview

In the Type 8 environment, application processes contain data that remote applications are able to read and write. The variable ASE defines the network visible attributes of application data and provides a set of services used to read, write, and report their values.

Individual variables are used to specify access to application data of any FAL-defined or user-defined data type. Services defined to access individual variables can be used to access one entire variables or one entry (per variable) in an array or one field (per variable) in a structure.

The formal model of the variable model is presented next, followed by a description of its services. IEC 61158-6-8 describes the abstract syntax and procedures for its protocol.

6.1.4.2 Requirements for access to variables

The structure of a variable value is defined by its data type. A variable's data type may be of any valid standardized or user defined data type. Standardized types and the facilities for defining user types are specified by the Data type Model.

6.1.4.3 Variable model class specification

6.1.4.3.1 Simple variable class specification

6.1.4.3.1.1 Simple variable formal model

FAL ASE:			**VARIABLE ASE**
CLASS:			**SIMPLE VARIABLE**
CLASS ID:			7
PARENT CLASS:			TOP
ATTRIBUTES:			
2	(m)	Attribute:	Data type
3	(m)	Attribute:	Length
5	(m)	Attribute:	Access Privilege
5.1	(m)	Attribute:	Password
5.2	(m)	Attribute:	Access Groups
5.3	(m)	Attribute:	Access Rights
6	(m)	Attribute:	Local Detail
SERVICES:			
1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
5	(o)	OpsService:	Information Report

6.1.4.3.1.2 Attributes

Data type

This attribute is the numeric identifier of a FIXED-LENGTH or STRING data type.

Length

This attribute indicates the length of simple variables data types. For variables with STRING data types, this attribute is used to define the maximum length of the variable in octets. For variables with the FIXED-LENGTH data type, this attribute is used to reflect the length of the variable as specified by the data type.

Access privilege

This attribute specifies the access controls defined for this variable. It is composed of the following:

Password

This attribute specifies the password for the access rights. Its value is null if it is not used.

Access groups

This attribute identifies which of the eight user-defined access groups are defined for the variable. More than one access group may be defined.

Access rights

This attribute defines the type of access defined for the variable. Valid values are

- Right to Write for Access Groups
- Right to Read for Access Groups
- Right to Write for the registered Password
- Right to Read for the registered Password
- Right to Write for all Communication Partners
- Right to Read for all Communication Partners.

Local detail

This attribute specifies local information.

6.1.4.3.1.3 Services

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

Read

This optional service may be used to read a single variable object or variable list object. This service may be used in both the client and the server model.

Write

This optional service may be used to update a single variable object or variable list object. This service may be used only in the client/server model.

Information report

This optional service is an unconfirmed service that may be used to report the value of a variable or variable list object. This service may be used in both the client/server model and the publisher/subscriber push model.

6.1.4.3.2 Array variable class specification

6.1.4.3.2.1 Formal model

FAL ASE:			**VARIABLE ASE**
CLASS:			**ARRAY VARIABLE**
CLASS ID:			8
PARENT CLASS:			TOP
ATTRIBUTES:			
2	(m)	Attribute:	Data type
2.1	(m)	Attribute:	Data type ID
3	(c)	Constraint:	Data type Format = FIXED-LENGTH | STRING
3.1	(o)	Attribute:	Element Length
5	(o)	Attribute:	Number of Elements
6	(m)	Attribute:	Access Privilege
6.1	(m)	Attribute:	Password
6.2	(m)	Attribute:	Access Groups

6.3	(m)	Attribute:	Access Rights
7	(m)	Attribute:	Local Detail

SERVICES:

1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
5	(o)	OpsService:	Information Report

6.1.4.3.2.2 Attributes

Data type

This attribute specifies the data type for elements of the array.

Element length

This conditional attribute indicates the length of an array element with an ARRAY, FIXED-LENGTH or STRING data type. For array elements with STRING data types, this attribute is used to define the length of an array element in octets. For variables with an ARRAY or FIXED-LENGTH data type, this attribute is used to reflect the length of an array element as specified by the data type.

Number of elements

This attribute specifies the number of array elements for variable objects that are defined as arrays. Array variables may be defined in one of two ways. Each of the ways, and the use of this attribute for each are shown below.

Method	Attribute Use
Reference to a data type with the format of ARRAY:	Reflects the number of elements defined for the array data type
Reference to a data type with the format of FIXED-LENGTH or STRING	Specifies the number of elements defined for the array variable

Access privilege

See the description of this attribute and its components defined in the Simple Variable class above.

Local detail

This attribute specifies local information.

6.1.4.3.2.3 Services

See the description of the services defined in the Simple Variable class above.

6.1.4.3.3 Record variable class specification

6.1.4.3.3.1 Formal model

FAL ASE:			**VARIABLE ASE**
CLASS:			**RECORD VARIABLE**
CLASS ID:			9
PARENT CLASS:			TOP

ATTRIBUTES:

2	(m)	Attribute:	Data type
3	(m)	Attribute:	Access Privilege
3.1	(m)	Attribute:	Password
3.2	(m)	Attribute:	Access Groups
3.3	(m)	Attribute:	Access Rights
4	(m)	Attribute:	List of Local Detail

SERVICES:

1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
5	(o)	OpsService:	Information Report

6.1.4.3.3.2 Attributes

Data type

This attribute specifies the data type for elements of the array.

Access privilege

See the description of this attribute defined for the Simple Variable class above.

Password

See the description of this attribute defined for the Simple Variable class above.

Access groups

See the description of this attribute defined for the Simple Variable class above.

Access rights

See the description of this attribute defined for the Simple Variable class above.

List of local detail

This attribute specifies the local detail for each element (field) of the record.

6.1.4.3.3.3 Services

See the description of the services defined for the Simple Variable class above.

6.1.4.4 Variable ASE service specification

6.1.4.4.1 Supported services

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

Read

Write

Information Report

The exact nature of the operation of variable services is determined, in part, by the application relationship over which they operate.

6.1.4.4.2 Read service

6.1.4.4.2.1 Service overview

This confirmed service may be used to read the value of a variable object. It may be used with application relationships configured to support the client/server model.

6.1.4.4.2.2 Service primitives

The service parameters for this service are shown in Table 14.

Table 14 – Read service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Variable Specifier	M	M (=)		
Numeric Identifier	S	S (=)		
Name	S	S (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
Value			M	M (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Variable specifier

This selector type parameter specifies the variable, variable list, or an element of an array or record variable.

Numeric identifier

This parameter identifies the variable by its Numeric Identifier (Index and Subindex).

Name

This parameter identifies the variable by its Name

Result(+)

This selection type parameter indicates that the service request succeeded.

Value

This parameter specifies the value read. For each of the variables, this parameter specifies the value of the variable. For variable lists, this parameter specifies the values of each of the variables in the list concatenated together in the order that they appear in the list. If any of the variables in a variable list could not be read, the service fails.

Result(-)

This selection type parameter indicates that the service request failed.

6.1.4.4.2.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

6.1.4.4.3 Write service

6.1.4.4.3.1 Service overview

This confirmed service is used to write the value of a variable.

6.1.4.4.3.2 Service primitives

The service parameters for this service are shown in Table 15.

Table 15 – Write service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Variable Specifier	M	M (=)		
Numeric Identifier	S	S (=)		
Name	S	S (=)		
Value	M	M (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

The argument contains the parameters of the service request.

Variable specifier

This parameter specifies the variable, or an element of an array or record variable.

Numeric identifier

This parameter identifies the variable by its Numeric Identifier (Index and Subindex).

Name

This parameter identifies the variable by its Name.

Value

This parameter specifies the value to write. For variables, this parameter specifies the value of the variable. For variable lists, this parameter specifies the values of each of the variables in the list concatenated together in the order that they appear in the list.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.1.4.4.3.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

6.1.4.4.4 Information report service

6.1.4.4.4.1 Service overview

This unconfirmed service is used by an application process to report the value of a variable to receiver(s) designated by the AR.

6.1.4.4.4.2 Service primitives

The service parameters for this service are shown in Table 16.

Table 16 – Information report service

Parameter name	Req	Ind
Argument		
AREP	M	M
Variable Specifier	C	C (=)
Numeric Identifier	S	S (=)
Name	S	S (=)
Value	M	M (=)

Argument

This parameter carries the parameters of the service invocation.

Variable specifier

This selector type parameter specifies the variable, or an element of an array or record variable.

Numeric identifier

This parameter identifies the variable by its Numeric Identifier (Index and Subindex).

Name

This parameter identifies the variable by its Name.

Value

This parameter specifies the value. For variables, this parameter specifies the value of the variable. For variable lists, this parameter specifies the values of each of the variables in the list concatenated together in the order that they appear in the list.

6.1.4.4.4.3 Service procedure

The Unconfirmed Service Procedure specified in 4.6 applies to this service.

6.1.5 Function invocation ASE

6.1.5.1 Overview

The FAL function invocation model defines two objects, the stateless action object and the state-oriented function invocation object.

Stateless function invocations, referred to as actions, run to completion when invoked and cannot be interrupted. In addition, some atomic actions return a value in response to being

invoked, while others do not. Those that do may be used to model software *functions*, and those that do not may be used to model software *procedures*.

State-oriented function invocations, on the other hand, may be controlled during their execution. Services are defined to suspend, resume, or abort them once they have been invoked. They do not return a value in response to being started. If it is necessary to abort a function invocation once it has been started, then it is defined as state-oriented.

State-oriented function invocations represent the network view of a specific invocation of a user function. If the user function can be invoked more than once simultaneously, then separate function invocation objects are needed to represent each invocation.

State-oriented function invocations may be used to model software processes or user operations that may be started and controlled.

NOTE An example of a state-oriented function invocation that may be used to model a user operation is the "playback" operation of a video recorder. Once it has been started, the playback operation may be stopped (paused) and later resumed.

To support the concept of software process modeling, the definition of state-oriented functional invocations includes optional attributes that can be used to relate them to load region objects.

The remainder of this subclause specifies the class definitions and services for the state-oriented *function invocation* object and the stateless *action* object.

6.1.5.2 Function invocation model class specifications

6.1.5.2.1 Function invocation class specification

6.1.5.2.1.1 Class overview

The function invocation class models the state-oriented function invocation. It may be used to model software processes or user functions whose operation may be controlled.

6.1.5.2.1.2 Formal model

FAL ASE:			**FUNCTION INVOCATION ASE**
CLASS:			**FUNCTION INVOCATION**
CLASS ID:			3
PARENT CLASS:			TOP
ATTRIBUTES:			
1	(m)	Attribute:	Access Privilege
1.1	(m)	Attribute:	Password
1.2	(m)	Attribute:	Access Groups
1.3	(m)	Attribute:	Access Rights
2	(m)	Attribute:	Deletable (TRUE,FALSE)
3	(m)	Attribute:	Reusable (TRUE,FALSE) Default : TRUE
4	(m)	Attribute:	Function Invocation State
54	(m)	Attribute:	Number of Related Objects In Use
SERVICES:			
1	(o)	OpsService:	Start
2	(o)	OpsService:	Stop
3	(o)	OpsService:	Resume
4	(o)	OpsService:	Reset

6.1.5.2.1.3 Attributes

Access Privilege

This attribute specifies the access controls defined for this function invocation. It is composed of the following:

Password

This attribute specifies the password for the access rights. Its value is null if it is not used.

Access groups

This attribute identifies which of the eight user-defined access groups are defined for the function invocation. More than one access group may be defined.

Access rights

This attribute defines the type of access defined for the function invocation. Valid values are

- Right to Delete for Access Groups(see note)
- Right to Start for Access Groups
- Right to Stop for Access Groups
- Right to Resume for Access Groups
- Right to Reset for Access Groups
- Right to Start for the Registered Password
- Right to Stop for the Registered Password
- Right to Resume for the Registered Password
- Right to Reset for the Registered Password
- Right to Start for all Communication Partners
- Right to Stop for all Communication Partners
- Right to Resume for all Communication Partners
- Right to Reset for all Communication Partners.

NOTE The right to delete has no meaning if the function invocation object is not deletable. This right requires the use of the Object Management Delete service, and not an operational service of the Function Invocation class.

Deletable

This attribute indicates, when TRUE, that the function invocation can be deleted using the Delete service. The value of this attribute is always TRUE for objects dynamically created with the Object Management ASE Create Service.

Reusable

This attribute indicates, when TRUE, that the function invocation may be executed more than once.

Function invocation state

This attribute indicates the current state of the function invocation. An enumerated set of values has been defined, and may be extended for local use within the AP to describe transient states of the Function Invocation. These transient state values for this attribute, however, are not visible over the network. The transitions between states and the events that cause transitions are user-defined.

UNRUNNABLE This state indicates that the function invocation is not executing and may not be executed.

IDLE This state indicates that the function invocation is not executing, but is capable of being executed.

RUNNING This state indicates that the function invocation is executing.

STOPPEDThis state indicates that the execution of a function invocation has been suspended.

STARTING This transient state indicates that the function invocation is being prepared for execution. Function invocations in this state are "ready to run".

STOPPING This transient state indicates that the execution of the function invocation is in the process of being stopped and its context saved.

RESUMING This transient state indicates that the function invocation is in the process of being returned to the executing state from a previously saved context.

RESETTING This transient state indicates that the function invocation is being reset to its initial context and removed from execution.

Number of related objects in use

This attribute indicates the number of related Load Region objects defined for this function invocation that are currently in the IN USE state. This attribute is incremented/decremented each time a related Load Region Object State Attribute goes/leaves the IN USE state. The Load Region State Machine gives the constraints for these transitions.

6.1.5.2.1.4 Services

Start

This optional service is used to request the function invocation to start executing. If this service is not present, it is assumed that the function invocation will be started using local means.

Stop

This optional service is used to request the function invocation to stop executing.

Resume

This optional service is used to request the function invocation to continue its execution.

Reset

This optional service is used to request the function invocation to return to its initial context and state.

6.1.5.2.2 Start service

6.1.5.2.2.1 Service overview

This confirmed service is used to request that a function invocation be started.

6.1.5.2.2.2 Service primitives

The service parameters for this service are shown in Table 17.

Table 17 – Start service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Key Attribute	M	M (=)		
Numeric Identifier	S	S (=)		
Name	S	S (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)
Function Invocation State			C	C (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Key attribute

This parameter specifies one of the key attributes of the function invocation object.

Numeric identifier

This parameter identifies the function invocation object by its Numeric Identifier (Index).

Name

This parameter identifies the function invocation object by its Name.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function invocation state

This conditional parameter specifies the current state of the function invocation. It is present if the error code indicates an object state conflict.

6.1.5.2.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

6.1.5.2.3 Stop service

6.1.5.2.3.1 Service overview

This confirmed service is used to stop a function invocation retaining its context so that it may be resumed.

6.1.5.2.3.2 Service primitives

The service parameters for this service are shown in Table 18.

Table 18 – Stop service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Key Attribute	M	M (=)		
Numeric Identifier	S	S (=)		
Name	S	S (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)
Function Invocation State			C	C (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Key attribute

This parameter specifies one of the key attributes of the function invocation object.

Numeric identifier

This parameter identifies the function invocation object by its Numeric Identifier (Index).

Name

This parameter identifies the function invocation object by its Name.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function invocation state

This conditional parameter specifies the current state of the function invocation. It is present if the error code indicates an object state conflict.

6.1.5.2.3.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

6.1.5.2.4 Resume service

6.1.5.2.4.1 Service overview

This confirmed service is used to resume the execution of a function invocation that has been stopped. Execution is resumed using the context saved when the function invocation was stopped.

6.1.5.2.4.2 Service primitives

The service parameters for this service are shown in Table 19.

Table 19 – Resume service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Key Attribute	M	M (=)		
Numeric Identifier	S	S (=)		
Name	S	S (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)
Function Invocation State			C	C (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Key attribute

This parameter specifies one of the key attributes of the function invocation object.

Numeric identifier

This parameter identifies the function invocation object by its Numeric Identifier (Index).

Name

This parameter identifies the function invocation object by its Name.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function invocation state

This conditional parameter specifies the current state of the function invocation. It is present if the error code indicates an object state conflict.

6.1.5.2.4.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service.

6.1.5.2.5 Reset service

6.1.5.2.5.1 Service overview

This confirmed service is used to reset a function invocation with its initial context. Its initial context is defined as the context of the function invocation set by its initialization procedures.

6.1.5.2.5.2 Service primitives

The service parameters for this service are shown in Table 20.

Table 20 – Reset service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Key Attribute	M	M (=)		
Numeric Identifier	S	S (=)		
Name	S	S (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)
Function Invocation State			C	C (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Invoke ID

This 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.

Key attribute

This parameter specifies one of the key attributes of the function invocation object.

Numeric identifier

This parameter identifies the function invocation object by its Numeric Identifier (Index).

Name

This parameter identifies the function invocation object by its Name.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function invocation state

This conditional parameter specifies the current state of the function invocation. It is present if the error code indicates an object state conflict.

6.1.5.2.5.3 Service procedure

The Confirmed Service Procedure specified In Clause 4 applies to this service.

6.1.5.3 Function invocation state machine

Table 21 gives the State Transitions for a Function Invocation object. The state “NON EXISTENT” indicates that the object does not exist. It is not defined as a valid value for the function invocation state attribute because attributes are created and deleted with the object (and therefore have no value when the object does not exist).

Table 21 – State transitions for a function invocation object

Transition	Current State	Events	Next State
5	IDLE	Start.Ind	STARTING
6	STARTING	Start succeeds: e.g. Related Load Region Object in state LOADED or IN-USE For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Start.Rsp(+)	RUNNING
7	STARTING	Start fails, non destructive: e.g. Related Load Region Object not in state LOADED or IN USE Start.Rsp(-)	IDLE
8	STARTING	Start fails, destructive Start.Rsp(-)	UN-RUNNABLE
9	RUNNING	Stop.Ind	STOPPING
10	STOPPING	Stop succeeds: e.g. Related Load Region Object in state IN USE For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Stop.Rsp(+)	STOPPED
11	STOPPING	Stop Execution fails, non destructive Stop.Rsp(-)	RUNNING
12	STOPPING	Stop fails, destructive For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Stop.Rsp(-)	UN-RUNNABLE
13	STOPPED	Resume.Ind	RESUMING
14	RESUMING	Resume succeeds: e.g. Related Load Region Object in state Loaded or In-Use For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Resume.Rsp(+)	RUNNING
15	RESUMING	Resume fails, non destructive: e.g. Load Region not in state LOADED or IN-USE Resume.Rsp(-)	STOPPED

Transition	Current State	Events	Next State
16	RESUMING	Resume fails, destructive For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Resume.Rsp(-)	UNRUNNABLE
17	STOPPED	Reset.Ind	RESETTING
18	RESETTING	Reset succeeds, reusable=TRUE Reset.Rsp(+)	IDLE
19	RESETTING	Reset succeeds, reusable = false Reset.Rsp(+)	UNRUNNABLE
20	RESETTING	Reset fails, non destructive Reset.Rsp(-)	STOPPED
21	RESETTING	Reset fails, destructive Reset.Rsp(-)	UNRUNNABLE
24	RUNNING	End of Program && Reusable=TRUE For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported)	IDLE
25	RUNNING	End of Program && Reusable = false For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported)	UNRUNNABLE
26	RUNNING	Program stopped For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported)	STOPPED

6.2 ARs

6.2.1 Queued user-triggered bi-directional with flow control (QUB-FC) AR endpoint class specification

6.2.1.1 Class overview

This class is defined to support asynchronous operation of the client/server model using queues. It provides for the conveyance of confirmed and unconfirmed services using application layer flow control for unconfirmed services and uses connection-oriented data link services for the exchanges. The data link priority for the transfers is specified separately for each transfer.

The behavior of this class is described as follows.

An AR ASE user wishing to convey a request APDU submits it 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 with exception of unconfirmed acknowledged services. The latter will be ignored if the outstanding service counter has reached the negotiated maximum.

The AREP receiving the request APDU from its underlying layer, queues it for delivery to its AR ASE user, in the order that it was received. If the received APDU contains an unconfirmed acknowledged request, the AREP issues the appropriated indication to the AR ASE user and a UCA_AckPDU to the underlying layer for transfer.

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 that it was received and decrements the appropriate flow control counter.

The following summarizes the characteristics of this AREP class.

Roles	Client Server Peer
Cardinality	1-to-1

6.2.1.2 Formal model

FAL ASE:			**AR ASE**
CLASS:			**Queued User-triggered Bi-directional with Flow Control AREP**
CLASS ID:			46
PARENT CLASS:			AR Endpoint
NETWORK MANAGEMENT ATTRIBUTES:			
1	(m)	Attribute:	Role (CLIENT, SERVER, PEER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	Remote Address Configuration Type (LINKED)
4	(m)	Attribute:	Maximum Outstanding Requests Calling
5	(m)	Attribute:	Maximum Outstanding Requests Called
6	(m)	Attribute:	Max Outstanding Unconfirmed Requests Client (maxUCSC)
7	(m)	Attribute:	Max Outstanding Unconfirmed Requests Server (maxUCSS)
8	(m)	Attribute:	Maximum Outstanding Requests Client (maxOSCC)
9	(m)	Attribute:	Maximum Outstanding Requests Server (maxOSCS)
10	(m)	Attribute:	Initiator (TRUE, FALSE)
11	(o)	Attribute:	Remote AP Name
12	(m)	Attribute:	Transmit DL Mapping Reference
13	(m)	Attribute:	Receive DL Mapping Reference
14	(m)	Attribute:	CIU
SERVICES:			
1	(o)	OpsService:	Confirmed Send
2	(o)	OpsService:	Unconfirmed Send
3	(o)	OpsService:	Establish
4	(o)	OpsService:	Abort

6.2.1.3 Network management attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PEER (CLIENT/SERVER) Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type indicate whether or not they are to be the initiator of the AR establishment process.

CLIENT Endpoints of this type issue confirmed and unconfirmed service Request-APDUs to servers and receive confirmed service Response-APDUs.

SERVER Endpoints of this type receive confirmed and unconfirmed service Request-APDUs from clients and issue confirmed service Response-APDUs to them. They may also issue unconfirmed service Request APDUs to clients.

AREP state

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-8.

Remote address configuration type

This attribute specifies how the remote address of the AREP is configured. The valid values are:

LINKED The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6-8).

Max outstanding requests calling

This attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP.

Max outstanding requests called

This attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user.

Max outstanding unconfirmed requests client (maxUCSC)

This attribute indicates the maximum number of requests that the AR Endpoint may be expecting from its local user.

Max outstanding unconfirmed requests server (maxUCSS)

This attribute indicates the maximum number of requests that the AREP may be expecting from the remote AREP.

Maximum outstanding requests client (maxOSCC)

This attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP.

Maximum outstanding requests server (maxOSCS)

This attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user.

Initiator

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR is the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of

this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

Remote AP name

This optional attribute specifies the name of the AP attached at the remote AREP.

Transmit DL mapping reference

This attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-8) are specified in IEC 61158-6-8.

Receive DL mapping reference

This attribute provides a reference to the underlying data-link layer mapping for the receive conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-8) are specified in IEC 61158-6-8.

CIU

This attribute (CIU - Control Interval User-triggered) contains the control interval for monitoring of a user triggered connection. The attribute is set by network management and used by the ARPM. If this attribute specifies a value not 0 then the ARPM transmits an Idle PDU while AR ASE user has not passed any service during the control interval (CIU/3). If CIU expires then the connections will be terminated.

6.2.1.4 Services

Confirmed send

This optional service is used to send a confirmed service on an AR.

Establish

This service is used to establish an AR.

Abort

This service is used to disconnect an AR.

Unconfirmed send

This service is used to send an unconfirmed service on an AR.

6.2.2 Buffered network-scheduled uni-directional (BNU) AR endpoint class specification

6.2.2.1 Class overview

This class is defined to support the “push” model for scheduled, 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 wishing to convey a request or response APDU submits it as an AR ASE Service Data Unit to its AREP for distribution. The AREP sending the request or response APDU writes it into the data-link layer buffer, completely replacing the existing contents of the buffer. The data-link layer transfers the buffer contents at the next scheduled transfer opportunity. The buffer is also transferred whenever a request to transmit is received, either locally or from the network.

NOTE Unscheduled requests to transmit have no effect on the behavior of this type of AR. They are treated as “out of band” with respect to the AR behavior.

If the AREP that sent the APDU receives another APDU before the buffer contents are transmitted, the buffer contents will be replaced with the new APDU, and the previous APDU

will be lost. When the buffer contents are transmitted, the AR ASE notifies the user of the transmission.

At the receiving endpoint, the APDU is received from the network and is immediately written into the buffer, completely 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 later request to receive the currently buffered APDU.

6.2.2.2 Formal model

FAL ASE:			**AR ASE**
CLASS:			**Buffered Network-Scheduled Uni-directional AR Endpoint**
CLASS ID:			38
PARENT CLASS:			AR Endpoint
NETWORK MANAGEMENT ATTRIBUTES:			
1	(m)	Attribute:	Role (PUSH PUBLISHER, PUSH SUBSCRIBER)
2	(m)	Attribute:	AREP State
2	(c)	Constraint	Role == PUSH SUBSCRIBER
3	(m)	Attribute:	DL Mapping Reference
SERVICES:			
1	(o)	OpsService:	Unconfirmed Send
3	(o)	OpsService:	Get Buffered Message

6.2.2.3 Network management attributes

Role

This attribute specifies the role of the AREP. The 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 published in confirmed service Response-APDUs.

AREP state

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-8.

DL mapping reference

For PUSH-PUBLISHER AREPs, this attribute specifies the mapping to the transmit conveyance path. For PUSH-SUBSCRIBER AREPs, this attribute specifies the mapping to the receive conveyance path. DL mapping attributes for the data-link layer (IEC 61158-3-8) are specified in IEC 61158-6-8.

6.2.2.4 Services

Unconfirmed send

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

Get buffered message

This local service is used to retrieve an APDU from the buffer used by an AR.

6.2.3 Queued user-triggered bi-directional transparent mode (QUB-TM) AR endpoint formal model

6.2.3.1 Description

This class is defined to support asynchronous operation of the publisher/subscriber model using queues. It provides a conveyance path for arbitrary data that are exchanged over a communication system.

The behavior of this class is described as follows.

An AR ASE user wishing to convey a request APDU submits it as an AR-Data-Send-Acknowledge Service to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity. The latter will be ignored if the outstanding service counter has reached the negotiated maximum.

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

The following summarizes the characteristics of this AREP class.

Roles	PUSH PUBLISHER PUSH SUBSCRIBER
Cardinality	1-to-1

6.2.3.2 Formal class definition

FAL ASE:			**AR ASE**
CLASS:			**Queued User-triggered Bi-directional Transparent Mode AREP**
CLASS ID:			50
PARENT CLASS:			AR Endpoint
NETWORK MANAGEMENT ATTRIBUTES:			
1	(m)	Attribute:	Role (PUSH Publisher, PUSH Subscriber)
2	(m)	Attribute:	Maximum Outstanding Requests Calling
3	(m)	Attribute:	Maximum Outstanding Requests Called
4	(m)	Attribute:	Transmit DL Mapping Reference
5	(m)	Attribute:	Receive DL Mapping Reference
SERVICES:			
1	(m)	OpsService:	AR-Data-Send-Acknowledge

6.2.3.3 Network management attributes

6.2.3.3.1 Role

This attribute specifies the role of the AREP. The valid values are:

PUSH PUBLISHER Endpoints of this type publish their data issuing AR-Data-Acknowledge service Request-APDUs.

PUSH SUBSCRIBER Endpoints of this type receive data published in AR-Data-Acknowledge service Request-APDUs.

6.2.3.3.2 Max outstanding requests calling

This attribute indicates the maximum number of requests that the AREP may be expecting from the remote AREP.

6.2.3.3.3 Max outstanding requests called

This attribute indicates the maximum number of requests that the AR Endpoint may be expecting from its local user.

6.2.3.3.4 Transmit DL mapping reference

This attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-8) are specified in IEC 61158-6-8.

6.2.3.3.5 Receive DL mapping reference

This attribute provides a reference to the underlying data-link layer mapping for the receive conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-8) are specified in IEC 61158-6-8.

6.2.3.4 Services

6.2.3.4.1 AR-data-send-acknowledge

This service is used to exchange the PDU on an AR.

6.3 Summary of FAL classes

This subclause contains a summary of the defined FAL Classes. The Class ID values have been assigned to be compatible with existing standards. Table 22 provides a summary of the classes.

Table 22 – FAL class summary

FAL ASE	Class	Class ID
Application Process	Application Process	16
Data type	Fixed Length & String Data type	5
	Structure Data type	6
Application Relationship	AREP	32
	BNU	38
	QUB-FC	46
	QUB-TM	50
Variable	Fixed Length & String Variable	7
	Array Variable	8
	Data Structure Variable	9
Function Invocation	Function Invocation	3

6.4 Permitted FAL services by AREP role

Table 23 below defines the valid combinations of services and AREP roles (which service APDUs and AREP with the specified role can send or receive). The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 23 – Services by AREP role

	Unonfirmedc	Confirmed	Client		Server		Push Publisher		Push Subscriber	
FAL Services			**req**	**rcv**	**req**	**rcv**	**req**	**rcv**	**req**	**rcv**
Mgt ASE										
Get Attributes		X	X			X				
AP ASE										
Identify		X	X			X				
Get Status		X	X			X				
Initiate		X	X			X				
Terminate		X	X			X				
Reject		X		X						
AR ASE										
AR-Unconfirmed Send							X			X
AR-Establish							X			X
AR-Abort							X		X	X
AR-Data-Send-Acknowledged							X			X
Variable ASE										
Read		X	X			X				
Write		X	X			X				
Information Report	X			X	X					
Function Invocation ASE										
Start		X	X			X				
Stop		X	X			X				
Resume		X	X			X				
Reset		X	X			X				

Bibliography

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

IEC 61158-2 (Ed.4.0), *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-3-8, *Industrial communication networks – Fieldbus specifications – Part 3-8: Data-link layer service definition – Type 8 elements*

IEC 61158-4-8, *Industrial communication networks – Fieldbus specifications – Part 4-8: Data-link layer protocol specification – Type 8 elements*

IEC 61158-6-8, *Industrial communication networks – Fieldbus specifications – Part 6-8: Application layer protocol specification – Type 8 elements*

IEC 61784-1 (Ed.2.0), *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61784-2, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

SOMMAIRE

AVANT-PROPOS 96
INTRODUCTION 98
1 Domaine d'application 99
1.1 Vue d'ensemble 99
1.2 Spécifications 100
1.3 Conformité 100
2 Références normatives 100
3 Termes et définitions 101
3.1 Termes de l'ISO/CEI 7498-1 101
3.2 Termes de l'ISO/CEI 8822 101
3.3 Termes de l'ISO/CEI 9545 101
3.4 Termes de l'ISO/CEI 8824 102
3.5 Termes liés à la couche liaison de données des bus de terrain 102
3.6 Définitions spécifiques à la couche application des bus de terrain 102
3.7 Abréviations et symboles 108
3.8 Conventions 108
4 Concepts 112
4.1 Vue d'ensemble 112
4.2 Relations architecturales 112
4.3 Structure de la couche application des bus de terrain 114
4.4 Dénomination et adressage de la FAL 127
4.5 Résumé de l'architecture 128
4.6 Procédures de services de la FAL 129
4.7 Procédures conceptuelles de services de la FAL [INFORMATIVE] 129
4.8 Attributs communs de la FAL 130
4.9 Paramètres de service communs de la FAL 130
4.10 Taille de l'APDU 131
5 ASE "Data type" 131
5.1 Généralités 131
5.2 Définition formelle des objets de types de données 133
5.3 Types de données définis de la FAL 134
5.4 Spécification de service de l'ASE "Data type" 137
6 Spécification du modèle de communication 138
6.1 ASE 138
6.2 AR 183
6.3 Résumé des classes de la FAL 189
6.4 Services de la FAL autorisés par le rôle de l'AREP 189
Bibliographie 191

Figure 1 – Relation avec le modèle de référence de base OSI 112
Figure 2 – Positionnement architectural de la FAL de Type 8 113
Figure 3 – Interactions Client/Serveur 116
Figure 4 – Interactions du modèle "push" 117
Figure 5 – Services d'APO transportés par la FAL 118
Figure 6 – Structure de l'entité d'application 120

Figure 7 – Exemple d'ASE de la FAL 122
Figure 8 – Gestion d'objets de la FAL 123
Figure 9 – Acheminement des services de l'ASE 124
Figure 10 – AREP définis et établis 127
Figure 11 – Composants architecturaux de la FAL 128
Figure 12 – Exemple de hiérarchie de la classe Data type 132
Figure 13 – L'ASE de l'AR achemine des APDU entre des AP 155
Figure 14 – Établissement des AR de 1 à 1 163

Tableau 1 – Paramètres du service Get attributes 139
Tableau 2 – Identify 147
Tableau 3 – Get status 148
Tableau 4 – Initiate 149
Tableau 5 – Terminate 152
Tableau 6 – Reject 153
Tableau 7 – Acheminement des primitives de service par le rôle de l'AREP 156
Tableau 8 – Combinaisons valides des rôles de l'AREP impliqués dans une AR 156
Tableau 9 – AR-unconfirmed send 161
Tableau 10 – Service AR-establish 162
Tableau 11 – Combinaisons valides des classes de l'AREP à lier 163
Tableau 12 – AR-abort 164
Tableau 13 – Paramètres de service AR-Data-Send-Acknowledge 166
Tableau 14 – Paramètres du service Read 171
Tableau 15 – Paramètres du service Write 172
Tableau 16 – Service Information Report 173
Tableau 17 – Paramètres du service Start 177
Tableau 18 – Paramètres du service Stop 178
Tableau 19 – Paramètres du service Resume 179
Tableau 20 – Paramètres du service Reset 180
Tableau 21 – Transitions d'état pour un objet Function Invocation 182
Tableau 22 – Résumé des classes de la FAL 189
Tableau 23 – Services par rôle de l'AREP 190

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-8: Définition des services de la couche application – Éléments de Type 8

AVANT-PROPOS

1) La CEI (Commission Électrotechnique Internationale) 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. À 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 des questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux 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. Toute divergence entre toute Publication de la CEI et toute publication nationale ou régionale correspondante doit être indiquée en termes clairs dans cette dernière.
5) La CEI n'a prévu aucune procédure de marquage valant indication d'approbation et n'engage pas sa responsabilité pour les équipements déclarés conformes à une de ses Publications.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition 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 brevets 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 la série CEI 61784. 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 61158-5-8 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.

Cette première édition et les parties qui l'accompagnent de la sous-série CEI 61158-5 annulent et remplacent la CEI 611585:2003. La présente édition de cette partie constitue une révision technique.

Cette édition de la CEI 61158-5 comporte les modifications significatives suivantes par rapport à l'édition précédente:

a) suppression de l'ancien bus de terrain de Type 6 à cause d'un manque d'adéquation avec le marché;

b) ajout de nouveaux types de bus de terrain;

c) partition de la partie 5 de la troisième édition en plusieurs parties numérotées -5-2, -5-3.

La présente version bilingue (2015-06) correspond à la version anglaise monolingue publiée en 2007-12.

Le texte anglais de cette norme est issu des documents 65C/475/FDIS et 65C/486/RVD.

Le rapport de vote 65C/486/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.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de maintenance 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 édition révisée, ou
- amendée.

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

La liste de toutes les parties de la série CEI 61158, publiées sous le titre général *Réseaux de communication industriels – Spécifications des bus de terrain*, peut être consultée sur le site web de la CEI.

INTRODUCTION

La présente partie de la CEI 61158 est l'une d'une série produite pour faciliter l'interconnexion de composants de systèmes d'automation. Elle est liée à d'autres normes dans l'ensemble tel que défini par le modèle de référence des bus de terrain "à trois couches" décrit dans le rapport technique CEI/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 des services d'application que les applications de bus de terrain et/ou la gestion de système peuvent exploiter.

Dans l'ensemble de normes relatives aux bus de terrain, le terme "service" désigne une fonctionnalité abstraite fournie par une couche du modèle de référence de base de l'interconnexion des systèmes ouverts (OSI, Open Systems Interconnection) 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 5-8: Définition des services de la couche application – Éléments de Type 8

1 Domaine d'application

1.1 Vue d'ensemble

La couche Application de bus de terrain (FAL, Fieldbus Application Layer) procure aux programmes de l'utilisateur un moyen d'accès à l'environnement de communication des bus de terrain. À cet égard, la FAL peut être vue comme une «fenêtre entre des programmes d'application correspondants».

La présente norme fournit des éléments communs pour les communications de messagerie à temps critique ou non-critique entre des programmes d'application dans un environnement d'automation et avec un matériel spécifique aux bus de terrain de Type 8. 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 niveau défini de certitude. La non-réalisation des actions spécifiées dans la fenêtre temporelle induit un risque de défaillance des applications qui demandent ces actions, avec les risques afférents pour l'équipement, les installations et éventuellement la vie humaine.

La présente norme définit de manière abstraite le service observable de l'extérieur fourni par la couche Application de bus de terrain de Type 8 en termes

a) d'un modèle abstrait pour définir des ressources (objets) d'application capables d'être manipulées par les utilisateurs par l'intermédiaire de l'utilisation 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, et

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

La présente norme vise à définir les services mis en place pour

1) l'utilisateur de la FAL, à la frontière entre l'utilisateur et la couche Application du modèle de référence de bus de terrain, et

2) la Gestion des systèmes, à la frontière entre la couche Application et la Gestion des systèmes selon le modèle de référence de bus de terrain.

La présente norme spécifie la structure et les services de la couche Application de bus de terrain de Type 8, 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, application entity) de la FAL contenues dans les processus d'application. L'AE de la FAL se compose d'un ensemble d'éléments de service d'application (ASE, Application Service Element) orientés objet et d'une entité de gestion de couche (LME, Layer Management Entity) qui gère l'AE. Les ASE fournissent des services de communication qui fonctionnent sur un ensemble de classes d'objets de processus d'application (APO, application process object) connexes. L'un des éléments ASE de la FAL est un ASE de gestion qui fournit un ensemble commun de services destinés à la gestion des instances des classes de la FAL.

Bien que ces services spécifient, du point de vue des applications, la manière dont la demande et les réponses sont émises et délivrées, ils n'incluent pas de spécification relative à ce que les applications qui demandent et qui répondent doivent en faire. En d'autres termes, les aspects comportementaux des applications ne sont pas spécifiés; seule une définition des demandes et réponses que ces applications peuvent envoyer/recevoir est établie. 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 commander certains aspects de son fonctionnement.

1.2 Spécifications

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

Un deuxième objectif est de fournir des trajets de migration à partir de protocoles préexistants de communications industrielles. C'est ce dernier objectif qui 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 spécification peut être utilisée comme la base pour les interfaces de programmation d'applications (Application Programming Interfaces) formelles. Néanmoins, elle n'est pas une interface de programmation formelle et il sera nécessaire pour toute interface de ce type de traiter de questions de mise en œuvre qui ne sont pas couvertes par la présente spécification, y compris

a) les tailles et l'ordonnancement des octets pour les divers paramètres de service à plusieurs octets, et
b) la corrélation de primitives appariées "request-confirm" (c'est-à-dire: demande et confirmation) ou "indication-response" (c'est-à-dire: indication et réponse).

1.3 Conformité

La présente norme ni ne spécifie de mises en œuvre individuelles ou de produits individuels ni ne contraint les mises en œuvre d'entités de couche application au sein des systèmes d'automation industriels.

Il n'y a pas de conformité d'équipement à la présente norme de définition de services de couche application. Au contraire, la conformité est obtenue par la mise en œuvre de protocoles conformes de couche Application qui satisfont aux services de couche Application de Type 8 tels que 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'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

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

ISO/CEI 646, *Technologies de l'information – Jeu ISO de caractères codés à 7 éléments pour l'échange d'informations*

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

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

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

ISO/CEI 8824, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN-1): Spécification de la notation de base*

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

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

3 Termes et définitions

Pour les besoins du présent document, les termes, définitions, symboles, abréviations et conventions suivants s'appliquent.

3.1 Termes de l'ISO/CEI 7498-1

La présente norme est basée en partie sur les concepts développés dans l'ISO/CEI 7498-1, et utilise les termes suivants lesquels y sont définis:

a) entité d'application
b) processus d'application
c) unité de donnée de protocole application
d) invocation de processus d'application
e) transaction d'application
f) système ouvert réel
g) syntaxe de transfert

3.2 Termes de l'ISO/CEI 8822

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 8822 s'appliquent:

a) syntaxe abstraite
b) contexte de présentation

3.3 Termes de l'ISO/CEI 9545

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 9545 s'appliquent:

a) association d'applications
b) contexte d'application
c) nom de contexte d'application
d) invocation d'entité d'application
e) type d'entité d'application
f) invocation de processus d'application
g) type de processus d'application
h) élément de service application

i) élément de service de contrôle d'application

3.4 Termes de l'ISO/CEI 8824

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 8824 s'appliquent:

a) identificateur d'objet

b) type

3.5 Termes liés à la couche liaison de données des bus de terrain

Pour les besoins du présent document, les termes suivants, tels que définis dans la CEI 61158-3-8 et la CEI 61158-4-8, s'appliquent.

a) DLCEP

b) DLSAP

c) Liaison

d) nœud

3.6 Définitions spécifiques à la couche application des bus de terrain

3.6.1
protection d'accès
limitation de l'usage d'un objet d'application à un seul client

3.6.2
allouer
prendre une ressource à partir d'une zone commune et affecter cette ressource à l'usage exclusif d'une entité spécifique

3.6.3
application
fonction ou structure de données pour laquelle des données sont consommées ou produites

3.6.4
interopérabilité de la couche application
capabilité des entités d'application à effectuer des opérations coordonnées et coopératives en utilisant les services de la FAL

3.6.5
objet d'applications
classes d'objets multiples qui gèrent et assurent un échange de messages pendant le mode exécution à travers le réseau et à l'intérieur de l'appareil de réseau

3.6.6
processus d'application
partie d'une application distribuée sur un réseau, située sur un appareil et associée à une adresse non ambiguë

3.6.7
identificateur de processus d'application
distingue de multiples processus d'application utilisés dans un appareil

3.6.8
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 ensemble de valeurs pour les attributs de leur classe (voir la définition de "Définition de classe d'objets de processus d'application"). Les définitions d'objet de processus d'application peuvent être accédées à distance à l'aide des services de l'ASE "Object Management" (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.

3.6.9
classe d'objets de processus d'application
classe d'objets de processus d'application définie en termes de l'ensemble de leurs attributs et services accessibles par le réseau

3.6.10
relation d'applications
association de type coopératif entre deux invocations d'entités d'application ou plus, dans un but d'échange d'informations et de coordination de leur action conjointe. Cette relation est activée soit par l'échange d'unités de données de protocole d'application, soit à la suite d'activités de pré-configuration

3.6.11
élément de service application de relation d'applications
élément de service d'application qui fournit le moyen exclusif d'établir et de terminer toutes les relations d'applications

3.6.12
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.

3.6.13
attribut
description d'une caractéristique ou d'une fonction, visible par un observateur externe, d'un objet

NOTE Les attributs d'un objet contiennent des informations relatives aux parties variables d'un objet. En règle générale, ils fournissent les informations de statut ou régissent le fonctionnement d'un objet. Les attributs peuvent également influer sur le comportement d'un objet. Les attributs se répartissent en deux catégories: les attributs de classe et les attributs d'instance.

3.6.14
comportement
indication de la manière dont un objet réagit à des événements particuliers

3.6.15
classe
un ensemble d'objets qui représentent tous la même sorte 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

3.6.16
attribut de classe
attribut qui est partagé par tous les objets au sein de la même classe

3.6.17
code de classe
identificateur unique affecté à chaque classe d'objets

3.6.18
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.

3.6.19
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

3.6.20
commandes de contrôle
invocations d'action transférées du client au serveur pour effacer des sorties, geler des entrées et/ou synchroniser des sorties

3.6.21
trajet d'acheminement
flux unidirectionnel d'unités APDU, d'un bout à l'autre d'une relation entre applications

Note 1 à l'article:L'abréviation «APDU» est dérivée du terme anglais développé correspondant «Application Protocol Data Unit».

3.6.22
cyclique
répétitif de manière régulière

3.6.23
cohérence de données
moyen pour une émission et un accès cohérents de l'objet de donnée d'entrée ou de sortie au sein du client et du serveur et entre eux

3.6.24
AR dédiée
AR utilisée directement par l'utilisateur de la FAL

NOTE Sur les AR dédiées, seulement l'en-tête de la FAL et les données utilisateur sont transférés.

Note 1 à l'article: L'abréviation «AR» est dérivée du terme anglais développé correspondant «Application relationship».

3.6.25
appareil
matériel physique connecté à la liaison

NOTE Un appareil peut contenir un ou plusieurs nœuds.

3.6.26
AR dynamique
AR qui nécessite l'utilisation de procédures d'établissement d'AR pour la placer dans un état établi

Note 1 à l'article: L'abréviation «AR» est dérivée du terme anglais développé correspondant «Application relationship».

3.6.27
nœud d'extrémité
noeud producteur ou consommateur

3.6.28
point d'extrémité
l'une des entités communicantes impliquées dans une connexion

3.6.29
ingénierie
terme abstrait qui caractérise l'application client ou l'appareil chargé(e) de configurer un système d'automation par l'intermédiaire d'éléments de données d'interconnexion

3.6.30
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)

3.6.31
classe d'erreur
regroupement général pour des définitions d'erreurs connexes et des codes d'erreurs correspondants

3.6.32
code d'erreur
identification d'un type spécifique d'erreur dans une classe d'erreurs

3.6.33
événement
instance d'un changement d'états

3.6.34
groupe
(a) collection d'objets

(b) adresse qui identifie plusieurs entités

3.6.35
interface
a) frontière partagée entre deux unités fonctionnelles, définies par les caractéristiques fonctionnelles, caractéristiques de signal ou autres caractéristiques selon le cas approprié

b) ensemble d'attributs et de services de classe de la FAL qui représente une vue spécifique sur la classe de la FAL

3.6.36
invocation
acte consistant à utiliser un service ou une autre ressource d'un processus d'application

NOTE Chaque invocation représente un fil (thread) de commande distinct 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.

3.6.37
index
adresse d'un objet au sein d'un processus d'application

3.6.38
instance
l'occurrence physique effective d'un objet au sein d'une classe permettant d'identifier un ou plusieurs objets au sein de la même classe d'objets

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

3.6.39
attributs d'instance
attribut qui est propre à une instance d'objet et n'est pas partagé par la classe d'objets

3.6.40
instancié
objet qui a été créé dans un appareil

3.6.41
ID de fabricant
identification de chaque fabrication de produit par un numéro unique

3.6.42
informations de gestion
informations accessibles par le réseau qui prennent en charge la gestion du fonctionnement du système de bus de terrain, y compris la couche application

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

3.6.43
membre
partie d'un attribut qui est structurée comme un élément d'une matrice

3.6.44
méthode
<objet> un synonyme pour un service opérationnel qui est fourni par l'ASE serveur et invoqué par un client

3.6.45
réseau
ensemble de nœuds reliés par un certain type de support de communication, notamment des répétiteurs intermédiaires, ponts, routeurs et passerelles de couches inférieures

3.6.46
objet
représentation abstraite d'un composant particulier au sein d'un appareil, habituellement un ensemble de données connexes (sous la forme de variables) et de méthodes (procédures) pour opérer sur les données en question qui ont une interface et un comportement clairement définis

3.6.47
service spécifique à un objet
service propre à la classe d'objets qui le définit

3.6.48
homologue
rôle d'un point d'extrémité de l'AR dans lequel il est capable d'agir en tant que client et serveur

3.6.49
connexion point-à-point
connexion qui existe entre exactement deux objets d'application

3.6.50
point d'extrémité prédéfini de l'AR
point d'extrémité de l'AR qui est défini localement à l'intérieur d'un appareil sans utilisation du service Create

NOTE Les AR prédéfinies qui ne sont pas préétablies sont établies avant leur utilisation.

3.6.51
point d'extrémité préétabli de l'AR
point d'extrémité de l'AR qui est mis dans un état établi pendant la configuration des AE qui commandent ses points d'extrémité

3.6.52
propriété
information descriptive relative à un objet

3.6.53
fournisseur
source d'une connexion de données

3.6.54
éditeur
rôle d'un point d'extrémité de l'AR qui émet des APDU sur le bus de terrain en vue de leur consommation par un ou plusieurs abonnés

NOTE Un éditeur peut ne pas être conscient de l'identité ou du nombre d'abonnés et il peut éditer ses APDU à l'aide d'une AR dédiée.

3.6.55
éditeur en poussée (ou éditeur push)
type d'éditeur qui édite un objet dans une APDU de demande de service non confirmé

3.6.56
abonné en poussée (ou abonné push)
type d'abonné qui reconnaît les APDU de demande de services non confirmés qu'il a reçues comme étant des données d'objet éditées

3.6.57
ressource
fonction de traitement d'informations d'un sous-système

3.6.58
serveur
a) rôle d'un AREP dans lequel il retourne au client qui a initié la demande une APDU de réponse de service confirmé

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

3.6.59
service
opération ou fonction qu'un objet et/ou une classe d'objets réalise à la demande d'un autre objet et/ou d'une autre classe d'objets

3.6.60
abonné
rôle d'un AREP dans lequel il reçoit des APDU produites par un éditeur

3.7 Abréviations et symboles

AE Application entity (Entité d'application)
AL Application layer (Couche d'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 donnée de protocole application)
API Application process identifier (Identificateur de processus d'application)
AR Application relationship (Relation d'applications)
AREP Application relationship endpoint (Point d'extrémité de relation d'applications)
ASE Application service element (Élément de service d'application)
CIM Computer integrated manufacturing (Production intégrée par ordinateur)
Cnf Confirmation
CR Communication relationship (Relation de communication)
CREP Communication relationship endpoint (Point d'extrémité de relation de communication)
DL- Data-link- (de couche liaison de données) (comme préfixe)
DLC Data-link connection (Connexion de liaison de données)
DLCEP Data-link connection endpoint (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 de liaison de données)
FAL Fieldbus application layer (Couche application des bus de terrain)
FIFO First in first out (premier entré, premier sorti)
ID Identifier (Identificateur)
Ind Indication
LME Layer management entity (Entité de gestion de couche)
OSI Open Systems Interconnect (Interconnexion des systèmes ouverts)
PDU Protocol data unit (Unité de données de protocole)
Req Request (Demande)
Rsp Response (Réponse)
SAP Service access point (Point d'accès au service)

3.8 Conventions

3.8.1 Vue d'ensemble

La couche FAL est définie comme étant un ensemble d'ASE orientés objet. Chaque ASE est spécifié dans un paragraphe distinct. Chaque spécification d'ASE est constituée de deux parties, à savoir sa spécification de classe et sa spécification de service.

La spécification de classe définit les attributs de la classe. Les attributs sont accessibles à partir des instances de la classe qui utilise les services de l'ASE Object Management

spécifiés dans l'Article 5 de la présente Norme. La spécification de service définit les services qui sont fournis par l'ASE.

3.8.2 Conventions générales

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

3.8.3 Conventions pour les définitions de classes

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

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

(1) La rubrique "FAL ASE:" est le nom de l'élément ASE de la couche FAL (FAL ASE) qui fournit les services pour la classe spécifiée.

(2) La rubrique "CLASS:" est le nom de la classe étant 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) La rubrique "CLASS ID:" est un numéro qui identifie la classe étant spécifiée. Ce numéro est unique au sein de l'ASE de la FAL qui fournira les services pour cette classe. Lorsqu'il est qualifié par l'identité de son ASE de la FAL, il identifie sans ambiguïté la classe relevant du domaine d'application de la FAL. La valeur "NULL" indique que la classe ne peut pas être instanciée. Les Class ID (identificateurs de classe) compris entre 1 et 255 sont réservés par la présente norme pour identifier des classes normalisées. Ils ont été affectés pour maintenir la compatibilité avec des normes nationales existantes. Les CLASS ID compris entre 256 et 2048 sont alloués pour identifier les classes définies par l'utilisateur.

(4) La rubrique "PARENT CLASS:" est le nom de la classe mère pour la classe étant spécifiée. Tous les attributs définis pour la classe mère 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 mère "TOP" indique que la classe étant définie est une définition de classe initiale. La classe mère "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" 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 dans la colonne 2: obligatoire (m) / facultatif (o) / conditionnel (c) / sélecteur (s), une étiquette de type d'attribut dans la colonne 3, un nom ou une expression conditionnelle dans la colonne 4, et, facultativement, une liste 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 un point (period). 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 (5). Les attributs peuvent être obligatoires (5.1) ou facultatifs (5.2) si la contrainte est vraie. Tous les attributs facultatifs n'exigent pas d'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 sont définis 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 (period). L'imbrication dans la liste de services sert à spécifier des services conditionnés à un énoncé de contrainte.

3.8.4 Conventions pour les définitions de service

3.8.4.1 Généralités

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

3.8.4.2 Paramètres de 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, il n'est pas indispensable d'énoncer tous les paramètres de façon explicite.

Les spécifications de service de la présente norme utilisent un format tabulaire pour décrire les paramètres de composants des primitives de services d'ASE. Les paramètres qui s'appliquent à chaque groupe de primitives de services sont définis dans des matrices. Chaque matrice comporte jusqu'à cinq colonnes:

1) le nom de paramètre,
2) la primitive "request",
3) la primitive "indication",
4) la primitive "response", et
5) la primitive "confirm".

Un paramètre (ou un composant de celui-ci) est énuméré dans chaque ligne de chaque matrice. 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, en fonction 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 par ailleurs qualifiées par des éléments entre parenthèses. Ces entrées peuvent être:

a) une contrainte spécifique à un paramètre:
"(=)" indique que le paramètre équivaut, du point de vue de la sémantique, au paramètre de la primitive de service située immédiatement à sa gauche dans la matrice;

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.

3.8.4.3 Procédures de services

Les procédures sont définies en termes

- d'interactions entre entités d'application par l'échange d'unités de données de protocole d'application de bus de terrain, et
- d'interactions entre un fournisseur de service de couche application et un utilisateur de service de couche application dans le même système par 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 communication à contrainte temporelle à l'intérieur de la couche Application de bus de terrain.

4 Concepts

4.1 Vue d'ensemble

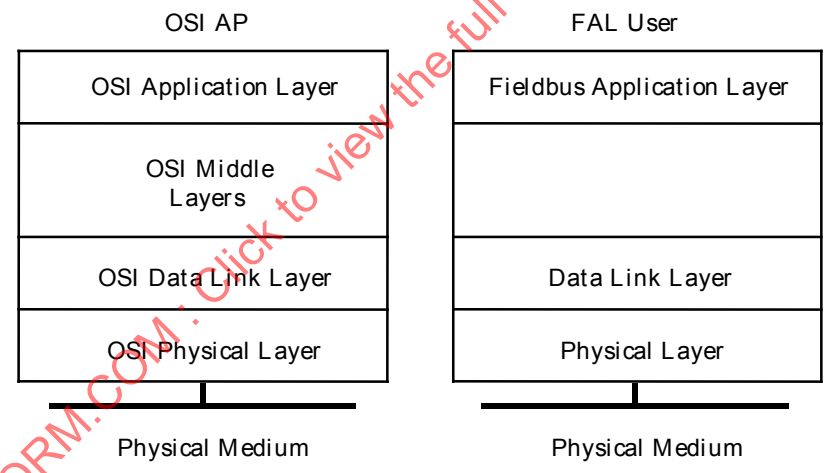
Le présent paragraphe décrit les caractéristiques fondamentales de la FAL. Des informations descriptives détaillées relatives à chacun des ASE de la FAL peuvent être consultées dans le paragraphe "Vue d'ensemble" de chacune des spécifications du Modèle de communication.

4.2 Relations architecturales

4.2.1 Relation avec la couche application du modèle de référence de base OSI

Les fonctions de la FAL ont été décrites selon les principes d'organisation en couches de l'OSI. Cependant, sa relation architecturale avec les couches inférieures est différente, comme montré à la Figure 1.

- La FAL inclut 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 une base pour spécifier la FAL;
- La FAL utilise directement les services de la couche sous-jacente. La couche sous-jacente peut être la couche liaison de données ou n'importe quelle couche intermédiaire. Lors de l'utilisation de la couche sous-jacente, la FAL peut fournir des fonctions normalement associées aux couches intermédiaires de l'OSI pour un mapping correct à la couche sous-jacente.



Légende

Anglais	Français
OSI AP	Processus d'application OSI
OSI application layer	Couche application OSI
OSI middle layers	Couches intermédiaires OSI
OSI data link layer	Couche liaison de données OSI
OSI physical layer	Couche physique OSI
FAL User	Utilisateur de la FAL
Fieldbus application layer	Couche application de bus de terrain
Data link layer	Couche liaison de données
Physical layer	Couche physique
Physical medium	Support physique

Figure 1 – Relation avec le modèle de référence de base OSI

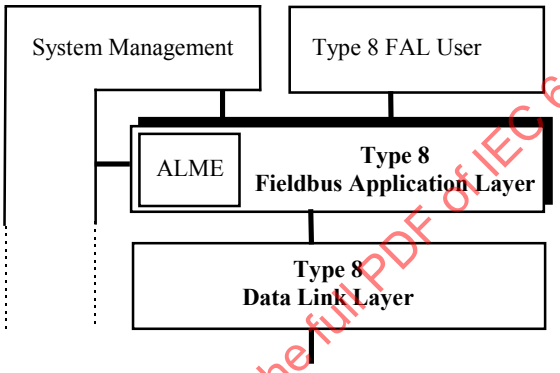
4.2.2 Relations avec d'autres entités de bus de terrain

4.2.2.1 Généralités

Les relations architecturales de la couche application des bus de terrain (FAL) de Type 8, telles qu'illustrées à la Figure 2, ont été conçues pour prendre en charge les besoins en interopérabilité des systèmes à temps critique répartis à l'intérieur de 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 des appareils de Type 8.

En outre, la FAL utilise directement la couche liaison de données pour transférer ses unités de données de protocole de couche application. Elle effectue cela en utilisant un ensemble de services de transfert de données et un ensemble de services d'appui utilisés pour commander les aspects opérationnels de la couche liaison de données.



Légende

Anglais	Français
System management	Gestion de système
Type 8 FAL User	Utilisateur de la FAL Type 8
Type 8 Fieldbus application layer	Couche application de bus de terrain Type 8
Type 8 Data link layer	Couche liaison de données Type 8

Figure 2 – Positionnement architectural de la FAL de Type 8

4.2.3 Mapping des capacités fonctionnelles

Deux mécanismes de communication sont pris en charge:

- émission cyclique de données d'une manière hautement efficace utilisant un modèle éditeur-abonné,
- émission acyclique de données utilisant un modèle de communication client-serveur

Les AREP agissant en tant qu'éditeur "push" ou en tant qu'abonné "push", sont utilisés pour l'émission cyclique.

Les AREP agissant en tant que client et serveur (homologue), sont utilisés pour l'émission de données acyclique. Une syntaxe de transfert spécifique est utilisée pour leurs APDU. Le mécanisme d'émission de données non cyclique est décrit par un sous-ensemble de ces ASE et services spécifiés en 6.1.

4.2.3.1 Utilisation de la couche liaison de données de Type 8

La couche application des bus de terrain (FAL) fournit aux AP de bus de terrain un accès au réseau. Elle s'interface directement à la couche liaison de données de bus de terrain pour le transfert de ses APDU.

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

4.2.3.2 Prise en charge d'applications de Type 8

Les applications de Type 8 sont représentées au réseau en tant que processus d'application (AP). Les AP sont les composants d'un système distribué qui peuvent être identifiés et accessibles individuellement.

Chaque AP contient une entité d'application (AE) de la FAL qui fournit l'accès au 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 en tant qu'objets de processus d'application (APO, Application Process Objects). 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 y accéder 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 utilisateur, les applications de blocs fonctionnels et de gestion.

4.2.3.3 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, les applications et le réseau de bus de terrain.

4.2.3.4 Accès aux entités de gestion de la couche FAL

Une seule entité de gestion de couche (LME, layer management entity) peut être présente dans chaque entité de la FAL sur le réseau. Les FALME fournissent un accès à la FAL pour des fins de gestion de système..

L'ensemble de données accessible par le gestionnaire de système est appelé Base d'informations de gestion de système (SMIB, system management information base). Chaque entité de gestion de la couche application des bus de terrain (FALME, Fieldbus Application Layer Management Entity) fournit la partie FAL de la SMIB. Comment la SMIB est mise en œuvre ne relève pas du domaine d'application de la présente norme.

4.3 Structure de la couche application des bus de terrain

4.3.1 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 TYPE 8.

La FAL diffère des autres couches OSI par deux aspects principaux:

- L'OSI définit un type simple de voie de communications de couche application, l'association, pour relier les AP les uns aux autres. La FAL définit la relation d'applications (AR, Application Relationship), dont il existe plusieurs types, permettant à des processus d'application (AP, Application Process) de communiquer les uns avec les autres;

- La FAL utilise la DLL pour transférer ses PDU 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. Entre la même paire (ou le même ensemble) de points d'accès au service de liaison de données, le protocole de FAL ne peut pas être utilisé concomitamment avec d'autres protocoles de couche application.

4.3.2 Concepts fondamentaux

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

La coopération entre les AP exige qu'ils partagent des informations suffisantes pour interagir et accomplir des activités de traitement d'une manière coordonnée. Ces activités peuvent être limitées à un seul segment de bus de terrain ou peuvent s'étendre sur de multiples 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 couche liaison de données (Partie 3 et Partie 4) peut fournir la distribution de temps à tous les appareils. Elles peuvent aussi définir des services d'appareils 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 fournissant un ensemble 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 suivants de la présente norme. La Partie 6 spécifie les protocoles nécessaires pour acheminer ces services d'objets entre les applications.

4.3.3 Processus d'application

4.3.3.1 Définition de l'AP

Dans l'environnement TYPE 8, une application peut être partitionnée en un ensemble de composants et répartie sur un certain nombre d'appareils présents sur le réseau. Chacun de ces composants est appelé "Processus d'application (AP)" de bus de terrain. Un AP de bus de terrain est une variation d'un processus d'application tel que défini dans le modèle de référence OSI de l'ISO (ISO/CEI 7498). Les AP de bus de terrain peuvent être associés à une adresse non ambigüe par au moins une adresse individuelle de point d'accès au service de la couche liaison de données. "Associé à une adresse non ambigüe", dans ce contexte signifie qu'aucun autre AP ne peut être localisé simultanément par la même adresse. Cette définition n'interdit pas qu'un AP soit situé par plusieurs adresses de point d'accès au service de liaison de données individuelles ou de groupe.

4.3.3.2 Services de communication

Les AP 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 Partie 6. 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.

Les services non confirmés, au contraire, 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 d'un un service confirmé.

4.3.3.3 Interactions entre AP

4.3.3.3.1 Généralités

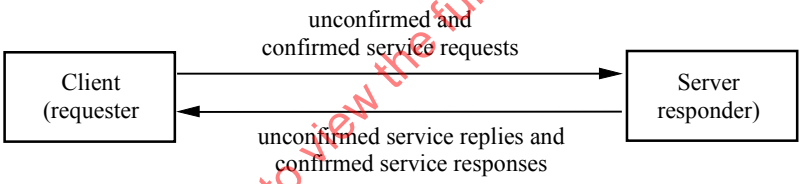
Les AP peuvent interagir avec d'autres AP selon les besoins, afin d'atteindre leurs objectifs fonctionnels. Aucune contrainte n'est imposée par la présente norme sur l'organisation de ces interactions ou les éventuelles relations qui peuvent exister entre elles.

Par exemple, 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 AP sont appelés "interactions client/serveur" et "interactions éditeur/abonné".

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

4.3.3.3.2 Interactions Client/Serveur

Les interactions client/serveur sont caractérisées par un flux bidirectionnel de données entre un AP client et un ou plusieurs AP serveurs. La Figure 3 illustre l'interaction entre un simple client et un simple serveur. Dans ce type d'interaction, le client peut émettre une demande de service confirmé ou non confirmé au serveur pour accomplir une certaine tâche. Si le service est confirmé, le serveur enverra toujours une réponse. Si le service est non confirmé, le serveur peut retourner une réponse à l'aide d'un service non confirmé défini à cet effet.



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

4.3.3.3.3 Interactions Éditeur/Abonné

4.3.3.3.3.1 Généralités

Les interactions éditeur/abonné, d'autre part, impliquent un simple AP éditeur et un AP abonné. Ce type d'interaction a été défini pour prendre en charge un modèle d'interactions entre AP, le modèle “push” (en poussée). Dans ce modèle, l'établissement de l'AP éditeur est effectué par la gestion et ne relève pas du domaine d'application de la présente norme.

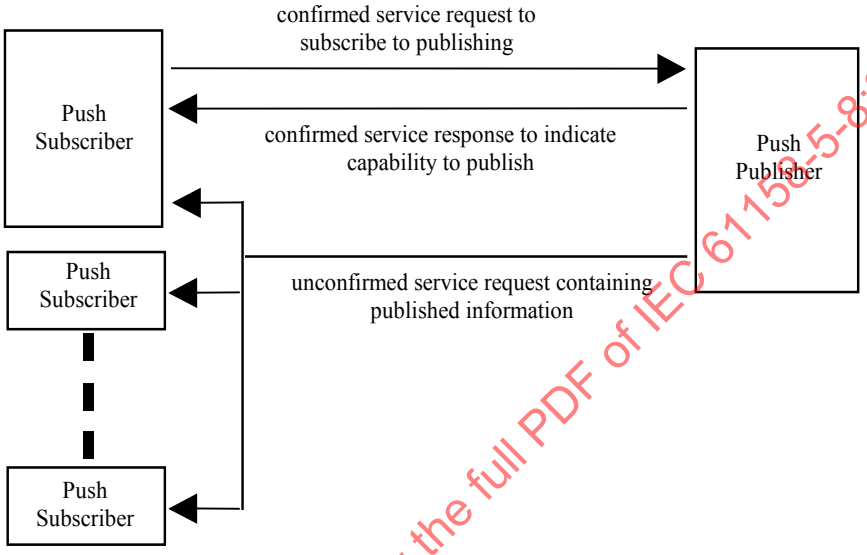
4.3.3.3.3.2 Interactions du modèle "push"

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 pour demander de joindre l'activité d'édition. 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 l'éditeur sont situés dans des AP différents.

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

Les abonnés pour le Modèle "Push" reçoivent les services non confirmés édités qui sont distribués par les éditeurs. La Figure 4 illustre le concept du Modèle "push" (en poussée).



Légende

Anglais	Français
Push subscriber	Abonné «push» (en poussée)
Push publisher	Éditeur «push» (en poussée)
Confirmed service request to subscribe to publishing	Demande de services confirmés pour s'abonner à l'édition
Confirmed service response to indicate capability to publish	Réponse de services confirmés pour indiquer la capacité d'éditer
Unconfirmed service request containing published information	Demande de services non confirmés contenant des informations éditées

Figure 4 – Interactions du modèle "push"

4.3.3.4 Structure de l'AP

Les données internes des AP peuvent être représentées par un ou plusieurs objets de processus d'application (APO, Application Process Objects) et accédées à travers une ou plusieurs entités d'application (AE, Application Entities). Les AE fournissent les moyens de communication de l'AP. Pour chaque AP, 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 la FAL.

4.3.3.5 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 la présente norme. Les classes définies par l'utilisateur peuvent aussi être spécifiées. Les identificateurs de classe (décrits en 3.8.2) sont attribués à partir d'un ensemble réservé à cet effet.

4.3.3.6 Type d'AP

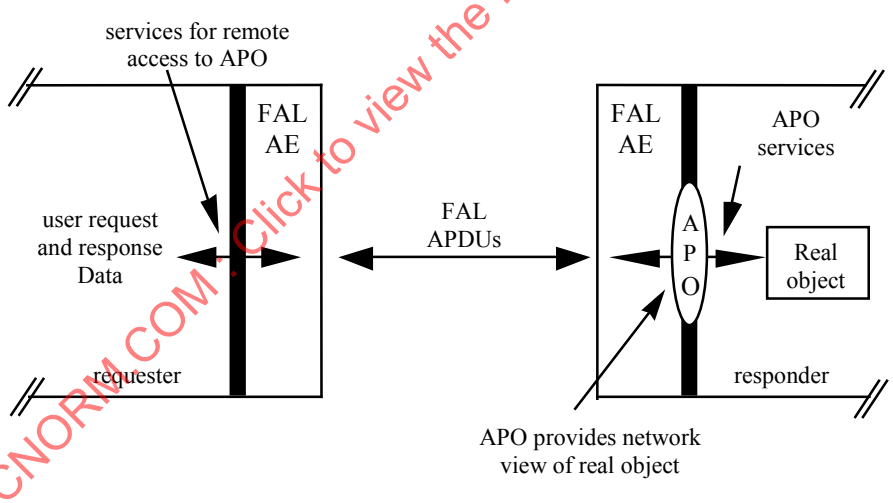
Comme décrit ci-dessus dans les paragraphes précédents, les AP sont définis par l'instanciation d'une classe d'AP. Chaque définition d'AP se compose des attributs et des services qui sont sélectionnés pour l'AP parmi ceux définis par sa classe d'AP. En outre, une définition d'AP contient des valeurs 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.

4.3.4 Objets de processus d'application

4.3.4.1 Définition de l'APO

Un objet de processus d'application (APO) est une représentation réseau d'un aspect spécifique d'un AP. Chaque APO représente un ensemble spécifique d'informations et de moyens de traitements d'un AP qui sont accessibles 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, comme montrés à la Figure 5, sont fournis par l'entité de communications de la FAL de l'AP, appelée entité d'application de la FAL (FAL AE, FAL Applications Entity).



Légende

Anglais	Français
Services for remote access to APO	Services pour un accès distant à l'APO
FAL AE	AE de la FAL
User request and response Data	Données de demande et de réponse de l'utilisateur
Requester	Demandeur
FAL APDUs	APDU de la FAL
APO services	Services d'APO
Real object	Objet réel
Responder	Répondeur
APO provides network view of real objects	L'APO fournit une vue réseau d'objets réels

Figure 5 – Services d'APO transportés par la FAL

Dans la Figure 5, 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 éditeur/abonné, les informations relatives à l'objet réel peuvent être éditées par l'intermédiaire de son APO. Les AP distants agissant comme abonnés observent la vue APO des informations éditées, au lieu d'avoir à connaître l'un quelconque des détails spécifiques à l'objet réel.

4.3.4.2 Classes d'APO

Une classe d'APO est une spécification générique pour un ensemble de plusieurs APO, chacun de ceux-ci étant décrit par le même ensemble d'attributs et accessible à l'aide du même ensemble 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 ensemble particulier d'attributs et de services d'AP accessibles par le réseau. La CEI 61158-6-8 spécifie la syntaxe et les procédures utilisées par le protocole de la FAL pour fournir un accès distant aux attributs et services d'une classe d'APO.

Les classes d'APO normalisées sont spécifiées par la présente norme pour le besoin 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 mère sont obligatoires pour la sous-classe. Les conventions pour définir les classes définies en 3.8.2 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.

4.3.4.3 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 par

(1) la sélection des attributs à partir de son modèle de classe d'APO qui doivent être accessibles à partir de l'objet réel;

(2) l'affectation des valeurs à un ou plusieurs attributs indiqués comme clés dans le modèle. Des attributs-clés sont utilisés pour identifier l'APO;

(3) l'affectation des valeurs à zéro, un, ou plusieurs attributs différents des attributs-clés pour l'APO. Des attributs différents des attributs-clés sont utilisés pour caractériser l'APO;

(4) la sélection des services dans le modèle qui peuvent être utilisés par les AP distants pour accéder l'objet réel.

Le paragraphe 3.8.2 de la présente Norme 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 qui indiquent si oui ou non l'attribut doit être présent dans un APO.

4.3.4.4 Types d'APO

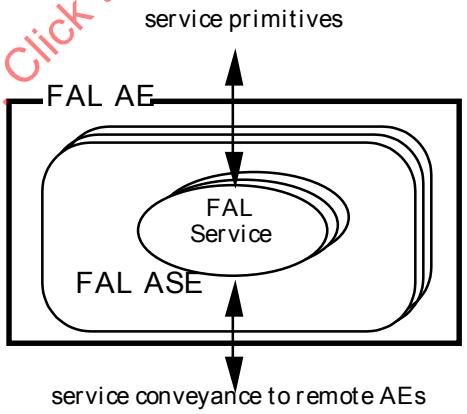
Les types d'APO fournissent le mécanisme pour définir des APO normalisés.

Comme décrit ci-dessus dans les paragraphes précédents, les APO sont définis par l'instanciation d'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 paramètres 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.

4.3.5 Entités d'application (AE, Application entities)

4.3.5.1 Définition de l'AE de la FAL

Une entité d'application fournit les moyens de communication pour un AP individuel. Une AE de la FAL fournit un ensemble 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 de service application (ASE) et, de ce fait, les services de FAL fournis à un AP sont définis par les ASE que contient son AE de la FAL. La Figure 6 illustre ce concept.



Légende

Anglais	Français
Service primitives	Primitives de services
FAL AE	AE de la FAL
FAL ASE	ASE de la FAL
FAL service	Service de la FAL
Service conveyance to remote AEs	Acheminent de services aux AE distantes

Figure 6 – Structure de l'entité d'application

4.3.5.2 Type d'AE

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

4.3.6 Éléments de service application

4.3.6.1 Généralités

Un élément de service application (ASE), tel que défini dans l'ISO/CEI 9545, est un ensemble de fonctions d'application qui fournissent un moyen pour l'interfonctionnement des invocations d'entités d'application dans un but spécifique. Les ASE fournissent un ensemble 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.

4.3.6.2 Services de la FAL

Les services de la FAL acheminent des demandes/réponses fonctionnelles entre les AP. Chaque service de la FAL est défini 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'Invoke ID (ID d'invocation) fourni par l'utilisateur. Cet Invoke ID est retourné dans la réponse par l'AP contenant l'objet réel. Lorsqu'il est présent, il peut être utilisé par l'utilisateur pour associer la réponse à la demande appropriée.

NOTE Cet Invoke ID spécifié par l'utilisateur est distinct de l'Instance ID que l'AP demandeur et son AE de la FAL peuvent eux-mêmes utiliser pour associer une primitive "request" à une primitive "confirm", une primitive "indication" à une primitive "response", ou une APDU de commande à une APDU de réponse. Voir 1.2.

Les services non confirmés peuvent être envoyés à partir 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.

4.3.6.3 Définition des ASE de la FAL

4.3.6.3.1 Généralités

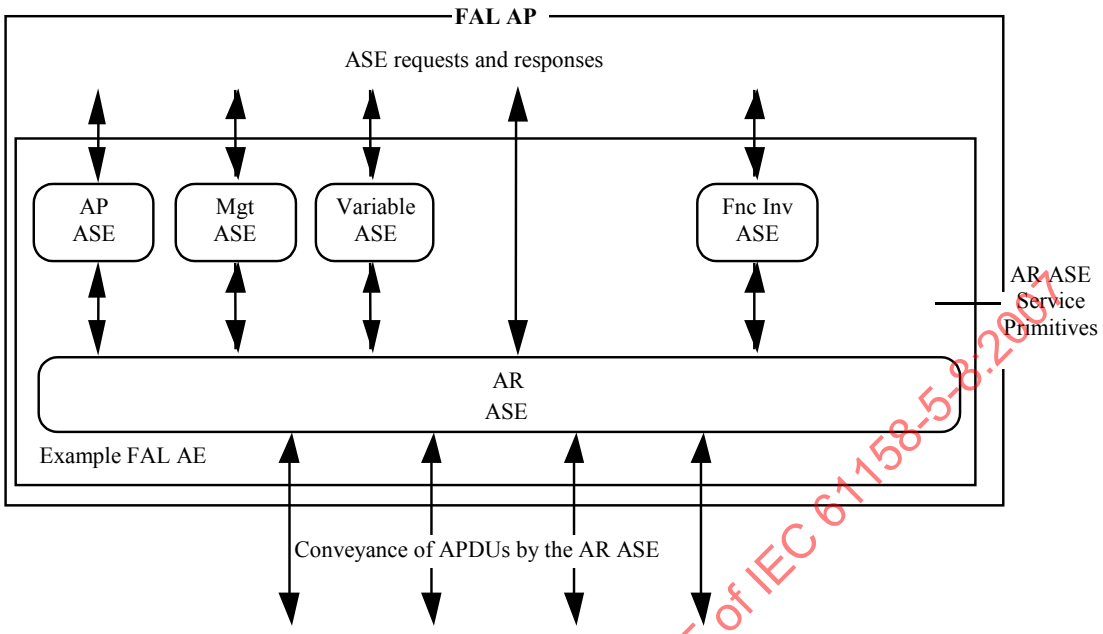
Une approche modulaire a été prise dans la définition des ASE de la FAL. Les ASE définis pour la FAL sont aussi orientés objet. En général, les ASE fournissent un ensemble de services conçus pour une classe d'objets spécifique ou pour un ensemble connexe de classes. Les ASE de gestion d'objets communs, lorsqu'ils sont présents, fournissent un ensemble commun de services de gestion applicables à toutes les classes d'objets.

Pour permettre l'accès à distance à l'AP, l'ASE de la relation d'applications est défini. Il fournit des services à l'AP pour définir et établir des relations de communication avec d'autres AP, et il fournit des services aux autres ASE pour acheminer leurs demandes de services et leurs réponses.

Chaque ASE de la FAL définit un ensemble de services, des APDU et des procédures qui fonctionnent sur les classes qu'il représente. Seul un sous-ensemble de services de l'ASE peut être fourni pour répondre aux besoins d'une application. Des profils peuvent être utilisés pour définir ces sous-ensembles. La définition de profils ne relève pas du domaine d'application de la présente norme.

Les APDU sont envoyées et reçues entre les ASE de la FAL qui prennent en charge les mêmes services. Chaque AE de la FAL contient, au minimum, l'ASE de l'AR et au moins un

autre ASE. La Figure 7 illustre un ensemble d'exemples des ASE de la FAL et leurs relations architecturales. Tous les ASE d'APO suivent cet exemple.



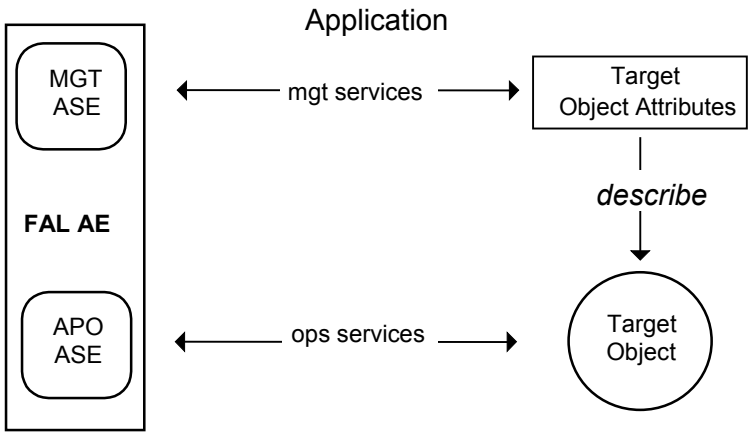
Légende

Anglais	Français
FAL AP	AP de la FAL
ASE requests and responses	Demandes et réponses de l'ASE
AP ASE	ASE d'AP
Mgt ASE	ASE de gestion
Variable ASE	ASE de variable
Fnc Inv ASE	ASE d'invocation de fonction
AR ASE Service Primitives	Primitives de services de l'ASE de l'AR
AR ASE	ASE de l'AR
Example FAL AE	Exemple d'une AE de la FAL
Conveyance of APDUs by the AR ASE	Acheminement des APDU par l'ASE de l'AR

Figure 7 – Exemple d'ASE de la FAL

4.3.6.3.2 ASE de gestion d'objets

Un ASE particulier 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
Application	Application
MGT ASE	ASE de gestion
FAL AE	AE de la FAL
APO ASE	ASO de l'APO
mgt services	Services de gestion
ops services	Services opérationnels
Target Object attributes	Attributs d'objet cible
describe	décrivent
Target object	Objet cible

Figure 8 – Gestion d'objets de la FAL

4.3.6.3.3 ASE de l'AP

Un ASE de l'AP peut être spécifié pour l'identification et le contrôle des AP de FAL. Les attributs définis par l'ASE de l'AP spécifient les caractéristiques de l'AP relatives à son fabricant, et énumèrent son contenu et ses fonctions.

4.3.6.3.4 ASE de l'APO

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

4.3.6.3.5 ASE de l'AR

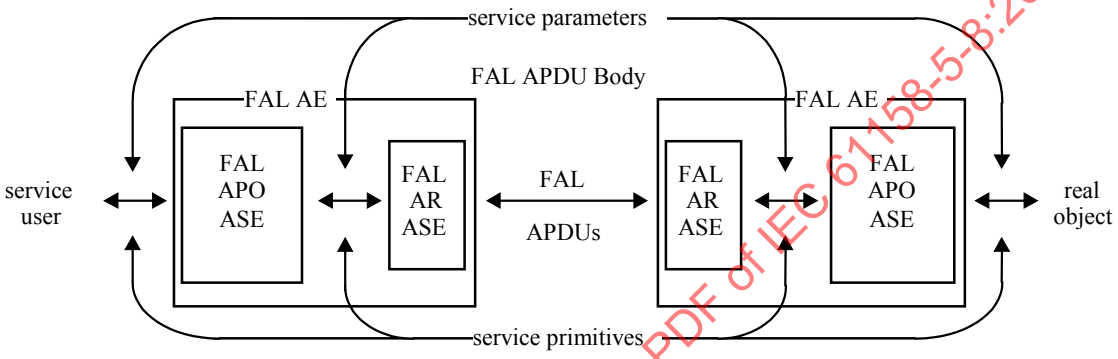
Un ASE de l'AR est spécifié pour établir et maintenir des relations d'applications (AR) qui sont utilisées pour acheminer des APDU de la FAL entre et parmi des AP. Les AR représentent des voies de communications de couche application entre des AP. Les ASE de l'AR sont chargés de fournir des services au niveau des points d'extrémité des AR. Les services d'es ASE de l'AR peuvent être définis pour établir, terminer et arrêter prématurément des AR, pour acheminer des APDU pour l'AE et pour indiquer à l'utilisateur le statut local de l'AR. En outre, les services locaux peuvent être définis pour accéder à certains aspects des points d'extrémité d'AR.

4.3.6.4 Acheminement des services de la FAL

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

Pour réaliser la tâche de transporter 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, ils acceptent les demandes de services et les réponses à transporter. En deuxième lieu, ils sélectionnent le type d'APDU de la FAL qui sera utilisé pour acheminer la demande ou la réponse et encodent les paramètres de service dans sa partie corps. Ensuite, ils présentent le corps d'APDU encodé à l'ASE de l'AR en vue de son acheminement.

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



Légende

Anglais	Français
Service parameters	Paramètres de services
FAL APDU Body	Corps d'APDU de la FAL
FAL AE	AE de la FAL
FAL APO ASE	ASE d'APO de la FAL
FAL AR ASE	ASE de l'AR de la FAL
FAL APDUs	APDU de la FAL
Service user	Utilisateur de service
Real object	Objet réel
Service primitives	Primitives de services

Figure 9 – Acheminement des services de l'ASE

4.3.6.5 Contexte de présentation de la 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 encoder chaque APDU. Cependant, l'architecture de communications des bus de terrain n'inclut pas la couche de présentation. Par conséquent, un mécanisme en variante est fourni pour la FAL par chacun des types spécifiques de modèles de communications.

4.3.7 Relations d'applications

4.3.7.1 Définition de l'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.

4.3.7.2 Points d'extrémité de l'AR

Les AR sont définies comme un ensemble d'AP coopératifs. L'ASE de l'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 de l'AR est utilisé par l'ASE de l'AR pour commander l'acheminement des APDU sur l'AR.

4.3.7.3 Classes de points d'extrémité de l'AR

Les AR sont constituées d'un ensemble de points d'extrémité de classes compatibles. Les classes de points d'extrémité de l'AR sont utilisées pour représenter les points d'extrémité de l'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.

4.3.7.4 Cardinalité de l'AR

Les AR caractérisent les communications entre les AP. L'une des caractéristiques d'une AR est le nombre de points d'extrémité de l'AR que contient l'AR. Les AR qui acheminent des services entre deux AP ont une cardinalité de 1 à 1. Accès à des objets par l'intermédiaire d'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 l'ensemble de services d'ASE qui peuvent être acheminés à destination et en provenance de ces objets par l'AR. La liste de services qui peuvent être acheminés par l'AR est choisie dans ceux définis pour l'AE.

4.3.7.5 Trajets d'acheminement de l'AR

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

4.3.7.6 Rôles de l'AREP

Puisque les AP interagissent les uns avec les autres par l'intermédiaire de 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 un client, un serveur, un éditeur ou un 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 d'initier 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 deux sens ou bien dans un seul sens.

4.3.7.7 Mémoires tampons et files d'attente de l'AREP

Les AREP peuvent être modélisés en file d'attente ou en mémoire tampon. 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 mémoire tampon est différent. Dans ce cas, une APDU devant être acheminée par l'ASE d'AR est placée dans une mémoire tampon pour le transfert. Lorsque la couche liaison de données obtient l'accès au réseau, elle émet le contenu de la mémoire tampon.

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

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

4.3.7.8 Acheminement déclenché par l'utilisateur et programmé

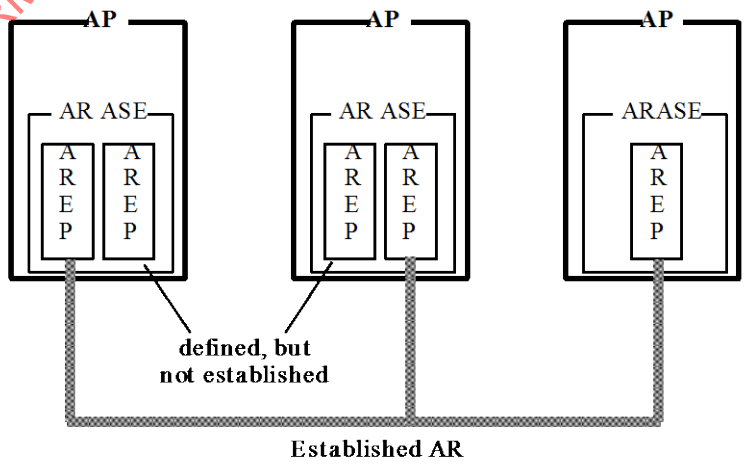
Une autre caractéristique d'un AREP est qu'il achemine les demandes de services et les réponses. Les AREP qui les transportent sur présentation par l'utilisateur sont appelés déclenchés par l'utilisateur. Leur acheminement est asynchrone par rapport au fonctionnement du réseau.

Les AREP qui transportent les demandes et les réponses à des intervalles prédéfinis, indépendamment de quand elles sont reçues pour le transfert sont qualifiés de «programmés».Des AREP programmés peuvent être capables d'indiquer l'instant où les données transférées ont été présentées tard pour l'émission ou l'instant où elles ont été présentées à temps, mais émises tard.

4.3.7.9 Définition et création d'AREP

Les définitions des AREP spécifient les instances de classes d'AREP. Les AREP peuvent être prédéfinis ou ils peuvent être définis à l'aide d'un service «Create» si leur AE prend en charge cette capabilité.

Les AREP peuvent être prédéfinis et préétablis, ou ils peuvent être prédéfinis et établis de façon dynamique. La Figure 10 décrit ces deux cas. Les AREP peuvent également exiger une définition et un établissement dynamiques ou ils peuvent être définis de manière dynamique de telle sorte qu'ils peuvent être utilisés sans établissement (ils sont définis dans un état établi).



Légende

Anglais	Français
AR ASE	ASE de l'AR

Anglais	Français
Defined, but not established	Définie, mais pas établie
Established AR	Relation d'applications établie

Figure 10 – AREP définis et établis

4.3.7.10 Établissement et terminaison 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 à travers l'échange des APDU d'AR.

Une fois qu'une AR a été établie, une AR peut être terminée progressivement ou elle peut être arrêtée prématurément, en fonction des fonctionnalités de l'AR.

4.4 Dénomination et adressage de la FAL

4.4.1 Généralités

Le présent paragraphe raffine les principes définis dans l'ISO 7498-3 qui impliquent l'identification (désignation) et l'emplacement (adressage) des APO référencés à travers la couche application des bus de terrain.

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

4.4.2 Identification des objets accessibles à travers la FAL

4.4.2.1 Généralités

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

Les identificateurs pour les AP et les APO au sein de la FAL sont définis comme des attributs-clés dans les définitions des classes pour les APO. Dans ces définitions d'APO, deux types d'attributs-clés sont communément utilisés: les noms et les identificateurs numériques.

4.4.2.2 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 du nom d'un APO soit spécifique à l'AP dans lequel il réside, l'attribution du nom est gérée au sein du système dans lequel il est configuré.

Les noms peuvent être descriptifs, toutefois ils n'ont pas à l'être. Les noms descriptifs permettent de donner des informations sensées telles que son utilisation, relatives à l'objet qu'ils désignent.

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

4.4.2.3 Identificateurs numériques

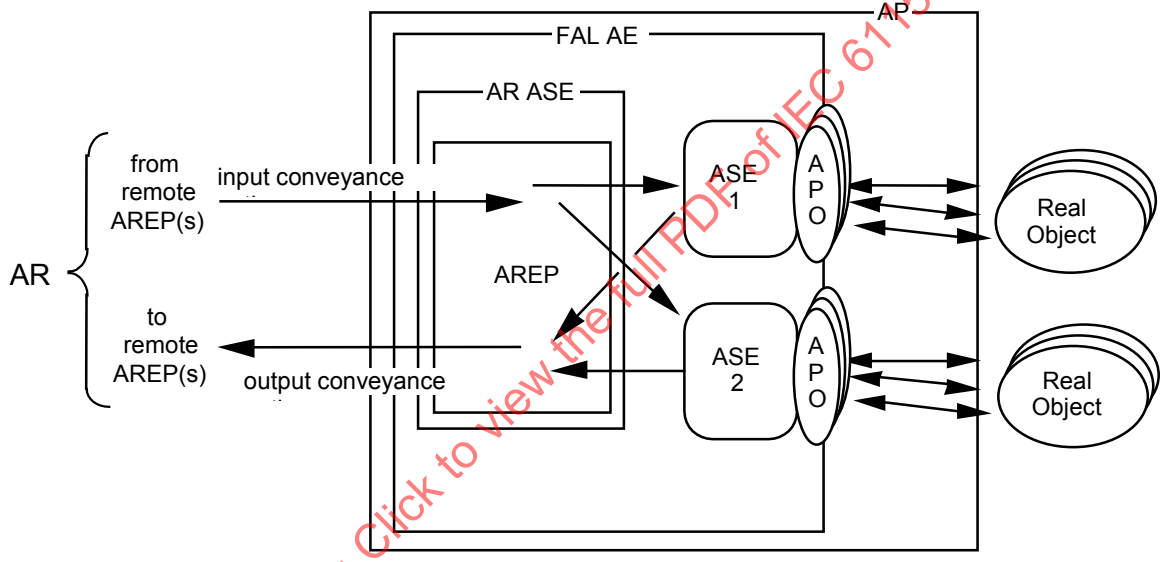
Les identificateurs numériques sont des identificateurs dont les valeurs sont des nombres. Ils sont conçus pour une utilisation efficace dans le système de bus de terrain, et peuvent être attribués pour un accès efficace à des APO par leur AP.

4.4.3 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.

4.5 Résumé de l'architecture

Le résumé de l'architecture de la FAL est présenté dans cet Article. La Figure 11 illustre les composants principaux de l'architecture de la FAL et la manière de les relier les uns aux autres.



Légende

Anglais	Français
FAL AE	AE de la FAL
AR ASE	ASE de l'AR
from remote AREP(s)	en provenance d'un ou plusieurs AREP distants
to remote AREP(s)	à destination d'un ou plusieurs AREP distants
Output conveyance path	Trajet d'acheminement en sortie
Input conveyance path	Trajet d'acheminement en entrée
Real object	Objet réel

Figure 11 – Composants architecturaux de la FAL

La Figure 11 décrit 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. Deux ASE sont montrés fournissant à leurs APO connexes les services d'accès à distance. L'ASE de l'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.

4.6 Procédures de services de la FAL

4.6.1 Procédures de services confirmés de la FAL

4.7 Procédures conceptuelles de services de la FAL [INFORMATIVE]

NOTE Le présent paragraphe décrit une méthode dans laquelle cette définition de services abstraits pourrait être réalisée dans une spécification et des mises en œuvre conformes de protocoles concrets. Voir 1.2.

4.7.1 Procédures conceptuelles de services confirmés de la FAL

L'utilisateur demandeur de service de la couche application invoque une primitive "request" de service confirmé de sa FAL. L'ASE approprié de la FAL crée un diagramme d'états de transactions pour commander l'invocation du service, attribue à ce diagramme d'états un InstanceID et un temps pour l'expiration de délai, construit le corps de l'APDU de demande de service confirmé connexe, y compris cet InstanceID et le transporte sur l'AR spécifiée.

Dès réception du corps de l'APDU de demande de service confirmé, l'ASE destinataire le décode. Si une erreur de protocole ne s'est pas produite, l'ASE destinataire crée un diagramme d'états de transactions pour gérer la réponse attendue, affecte à ce diagramme d'états un (deuxième) InstanceID indépendant, puis délivre à son utilisateur de service AL une primitive "indication" de service confirmé, avec le (deuxième) InstanceID comme un paramètre de mise en œuvre supplémentaire.

Si l'utilisateur de service AL répondeur est en mesure de traiter la demande avec succès, l'utilisateur retourne une primitive "response (+)" de service confirmé, identifiant la transaction par l'InstanceID présenté comme faisant partie de la primitive "indication" stimulante.

Si l'utilisateur répondeur n'est pas en mesure de traiter la demande avec succès, le service échoue et l'utilisateur émet une primitive "response (–)" de service confirmé indiquant la raison de l'échec, identifiant de nouveau la transaction par l'InstanceID présenté comme faisant partie de la primitive "indication" stimulante.

Quelle que soit la réponse choisie par les utilisateurs de services AL, l'ASE destinataire rend disponibles, et les informations à partir de la primitive "response", et les informations à partir de la primitive "indication" associée lorsqu'il forme l'APDU qui doit être retournée à l'ASE émetteur.

L'ASE répondeur construit un corps de l'APDU de réponse de service confirmé pour une primitive "response (+)" de service confirmé ou un corps de l'APDU d'erreur de service confirmé pour une primitive response (–) de service confirmé, chacun contenant le (premier) InstanceID de l'APDU de demande initiale, et la transporte sur l'AR spécifiée.

Dès réception du corps de l'APDU de réponse ou d'erreur, l'ASE émetteur utilise le (premier) InstanceID contenu dans l'APDU de réponse ou d'erreur afin de faire associer l'APDU au diagramme d'états et à la demande appropriés. Une fois que cette association a été réalisée, l'ASE émetteur rend disponibles, et les informations à partir de l'APDU reçue et les informations à partir de la primitive "request" associée. Il envoie une primitive "confirmation" de service confirmé à l'ASE demandeur de la FAL qui spécifie s'il s'agit d'un succès ou d'un échec, indique la raison de l'échec en cas d'échec, et annule le diagramme d'états de transactions associé.

Si le temporisateur associé au diagramme d'états expire avant que l'ASE émetteur reçoive l'APDU de réponse ou d'erreur retournée, l'ASE de l'AR envoie une primitive "confirmation(–)" de service confirmé à l'ASE demandeur de la FAL et annule le diagramme d'états de transactions associé.

4.7.2 Procédures conceptuelles de services non confirmés de la FAL

L'utilisateur demandeur présente une primitive "request" de service non confirmé à son AE de la FAL. L'ASE approprié de la FAL construit le corps correspondant d'APDU de demande de service non confirmé et l'achemine sur l'AR spécifiée.

À la réception du corps d'APDU de demande de service non confirmé, l'ASE ou les ASE destinataire(s) participant à l'AR délivre(nt) à son/leur utilisateur la primitive "indication" de service non confirmé appropriée. .

4.8 Attributs communs de la FAL

Dans les spécifications des classes de FAL qui suivent, plusieurs classes utilisent les attributs suivants. Par conséquent, ces attributs sont définis ici, au lieu de l'être avec les autres attributs pour chacune des classes, à l'exception de la classe Data type.

ATTRIBUTES:

1 (o) Key Attribute: Numeric Identifier

2 (o) Key Attribute: Name

Numeric identifier

Cet attribut-clé facultatif spécifie l'identificateur numérique de l'objet. Il est utilisé comme une référence sténographique par le protocole de la FAL pour identifier l'objet. Il existe trois possibilités pour des fins d'identification: identificateur numérique ou nom ou les deux. Cet attribut est requis pour le modèle Data type (type de données).

Name

Cet attribut-clé facultatif spécifie le nom de l'objet. Il existe trois possibilités pour des fins d'identification: identificateur numérique ou nom ou les deux.

4.9 Paramètres de service communs de la FAL

Dans les spécifications des services de la FAL qui suivent, plusieurs services utilisent les paramètres suivants. Par conséquent, ils sont définis ici, au lieu de l'être avec les autres paramètres pour chacun des services.

AREP

Ce paramètre spécifie des informations suffisantes pour identifier localement l'AREP qui doit 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

Ce paramètre fourni par l'utilisateur identifie cette invocation du service. Il est utilisé pour associer des demandes de services à des réponses. Par conséquent, il convient que deux invocations de services en cours ne soient pas identifiées par la même valeur d'Invoke ID

NOTE 2 Le mécanisme par lequel une mise en œuvre de ces services abstraits corrèle des primitives "request" et "confirm" au niveau d'un AREP, des primitives "indication" et "response" connexes au niveau d'un deuxième AREP, et les APDU qui lient des AE distants pour mettre en œuvre ces services, est une question de protocole comme énoncé en 1.2. Un Instance ID d'un service pouvant être transporté dans ces APDU est logiquement non lié à cet Invoke ID fourni par l'utilisateur.

FAL ASE/FAL class

Ce paramètre spécifie l''ASE de la FAL (par exemple AP, AR, Variable, Data type, Event, Function Invocation, Load Region) et la classe de la FAL dans l'ASE (par exemple AREP, Variable List, Notifier, Action).

Numeric ID

Ce paramètre est l'identificateur numérique de l'objet.

Error info

Ce paramètre fournit des informations d'erreur relatives aux erreurs de service. Il est retourné dans des primitives "response(-)" de service confirmé. Il se compose des éléments suivants.

Error class

Ce paramètre indique la classe générale d'erreur. Les valeurs valides sont spécifiées dans la définition du paramètre Error Code ci-dessous.

Error code

Ce paramètre identifie l'erreur de service spécifique.

Additional code

Ce paramètre facultatif identifie l'erreur produite lors du traitement de la demande 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"

4.10 Taille de l'APDU

La taille de l'APDU est fonction du modèle de communication.

5 ASE "Data type"

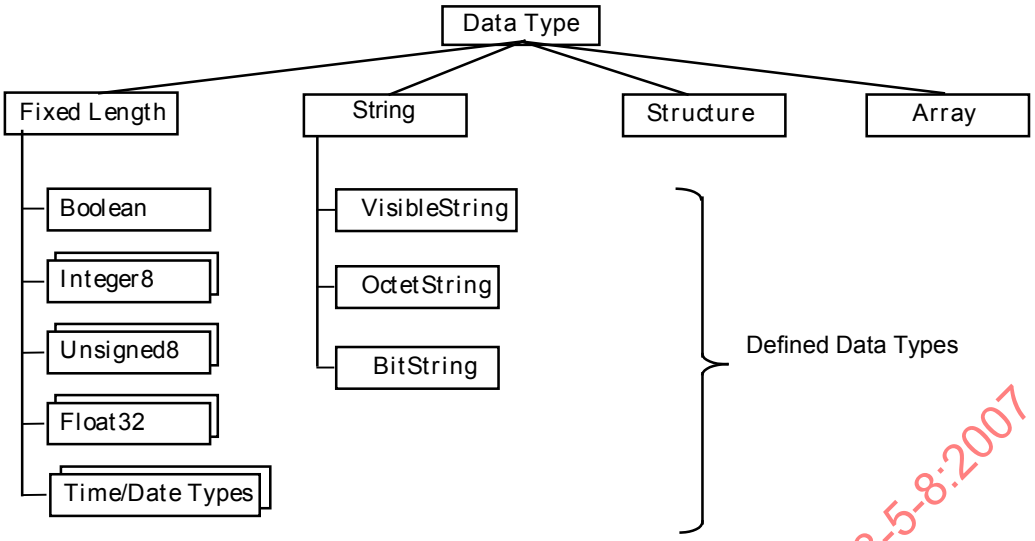
5.1 Généralités

5.1.1 Vue d'ensemble

Les types de données spécifient la syntaxe indépendante de toute machine pour les données d'application acheminées par les services de la FAL. La couche application des bus de terrain prend en charge la définition et le transfert des types de données tant de base que construits. Les règles d'encodage pour les types de données spécifiés dans le présent Article sont fournies dans la CEI 61158-6-8.

Les types de base sont des types atomiques qui ne peuvent pas être décomposés en des types plus élémentaires. Les types construits sont des types composés de types de base et d'autres types construits. Leur complexité et leur profondeur d'imbrication ne sont pas contraintes par la présente Norme.

Les types de données sont définis comme des instances de la classe de types de données, comme montré à la Figure 12. 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.



Légende

Anglais	Français
Defined data types	Types de données définis
Data types	Types de données

Figure 12 – Exemple de hiérarchie de la classe Data type

Les définitions de types de données dans la Figure 12 sont représentées sous la forme d'une structure classe/format/instance commençant par la classe de types de données intitulée "Data type". Les formats pour les types de données sont définis par la classe de types de données et sont représentés à la Figure 12.

Les classes de données de base sont utilisées pour définir des types de données Fixed Length (longueur fixe) et Bitstring (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 et les structures.

5.1.2 Présentation des 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 aux 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 classes 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 en 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 Présentation des types de longueur fixe

La longueur des types Fixed length est un nombre entier d'octets.

5.1.4 Présentation des types Constructed

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 n'ont pas à être du même type.

5.1.4.4 Niveau d'imbrication

Seul le niveau 1 d'imbrication est utilisé.

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 des données utilisateur

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

Les règles pour coder les données utilisateur dans les unités de données du protocole de FAL sont fonction du type de données. Ces règles sont définies dans la CEI 61158-6-8. Les types de données définis par l'utilisateur pour lesquels il n'y a pas de règles d'encodage sont transférés sous la forme d'une séquence 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 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 mère "top" indique le sommet de l'arbre de la classe de la FAL.

FAL ASE: **DATA TYPE ASE**

CLASS:			**DATA TYPE**
CLASS ID:			5 (FIXED LENGTH & STRING), 6 (STRUCTURE)
PARENT CLASS:			TOP
ATTRIBUTES:			
1	(m)	Key Attribute:	Data type Numeric Identifier
2	(o)	Key Attribute:	Data type Name

5.2.1.2 Attributs

Data type Numeric Identifier

Cet attribut identifie l'identificateur numérique du type de données correspondant.

Data type Name

Cet attribut facultatif identifie le nom du type de données correspondant.

5.3 Types de données définis de la FAL

5.3.1 Types Fixed length (longueur fixe)

5.3.1.1 Types Boolean (booléens)

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

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

5.3.1.2 Types de données

5.3.1.2.1 BinaryDate

CLASS:			**Type de données**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	11
2	Data type Name	=	BinaryDate
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

Ce type de données est constitué de six éléments de valeurs unsigned (non signées) et exprime l'heure et la date calendaires. Le premier élément est un type de données Unsigned16 et donne la fraction d'une minute en millisecondes. Le deuxième élément est un type de données Unsigned8 et donne la fraction d'une heure en minutes. Le troisième élément est un type de données Unsigned8 et donne la fraction d'un jour en heures. Le quatrième élément est un type de données Unsigned8. Ses trois (3) bits de poids fort donnent le jour de la semaine et ses cinq (5) bits de poids faible donnent le jour du mois. Le cinquième élément est un type de données Unsigned8 et donne le mois. Le dernier élément est un type de données Unsigned8 et donne l'année.

5.3.1.2.2 BinaryDate2000

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	51
2	Data type Name	=	BinaryDate2000
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

Ce type de données est constitué de six éléments de valeurs unsigned (non signées) et exprime l'heure et la date calendaires. Le premier élément est un type de données Unsigned16 et donne la fraction d'une minute en millisecondes. Le deuxième élément est un type de données Unsigned8 et donne la fraction d'une heure en minutes. Le troisième élément est un type de données Unsigned8 et donne la fraction d'un jour en heures. Le quatrième élément est un type de données Unsigned8. Ses trois (3) bits de poids fort donnent le jour de la semaine et ses cinq (5) bits de poids faible donnent le jour du mois. Le cinquième élément est un type de données Unsigned8 et donne le mois. Le dernier élément est un type de données Unsigned16 et donne l'année.

5.3.1.2.3 TimeOfDay

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
4	Format	=	FIXED LENGTH
4.1	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 et donne l'heure après minuit en millisecondes. Le deuxième élément est un type de données Unsigned16 et donne la date en comptant les jours à partir du 1er janvier 1984.

5.3.1.2.4 TimeDifference

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	13
2	Data type Name	=	TimeDifference
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4 ou 6

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

5.3.1.3 Types numériques

5.3.1.3.1 Floating Point types (virgule flottante) – Float32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float32
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Ce type a une longueur de quatre octets. Le format de float32 est celui défini par l'ISO 60559 comme étant en simple précision ("single precision").

5.3.1.3.2 Types Integer (entiers)

5.3.1.3.2.1 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

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

5.3.1.3.2.2 Integer16

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	3
2	Data type Name	=	Integer16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

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

5.3.1.3.2.3 Integer32

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.3.3 Types non signés (Unsigned)

5.3.1.3.3.1 Unsigned8

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.3.3.2 Unsigned16

CLASS: **Data type**

ATTRIBUTES:

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

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type Unsigned a une longueur de deux octets.

5.3.1.3.3.3 Unsigned32

CLASS: **Data type**

ATTRIBUTES:

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

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type Unsigned a une longueur de quatre octets.

5.3.2 Types String (chaîne)

5.3.2.1 BitString

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	14
2	Data type Name	=	Bitstring
3	Format	=	STRING
5.1	Octet Length	=	1 à n

Ce type String (chaîne) est défini comme une série d'éléments BitString8.

5.3.2.2 OctetString

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	10
2	Data type Name	=	OctetString
3	Format	=	STRING
4.1	Octet Length	=	1 à n

OctetString est une séquence ordonnée d'octets, numérotée de 1 à n. Pour les besoins de discussion, l'octet 1 de la séquence est désigné comme étant le premier octet. L'ordre d'émission est défini dans la CEI 61158-6-8.

5.3.2.3 VisibleString

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	9
2	Data type Name	=	VisibleString
3	Format	=	STRING
4.1	Octet Length	=	1 à n

Ce type est défini comme étant le type "string" (chaîne) de l'ISO/CEI 646.

5.4 Spécification de service de l'ASE "Data type"

Il n'y a pas de services opérationnels définis pour l'objet de type.

6 Spécification du modèle de communication

6.1 ASE

6.1.1 ASE de gestion d'objets

6.1.1.1 Vue d'ensemble

Les modèles de l'ASE de la FAL spécifient chacun une ou plusieurs classes connexes d'APO de la FAL comme un ensemble d'attributs et de services. Les services de la FAL définis pour une classe sont utilisés pour manipuler, contrôler des APO de la classe ou y accéder. Les services qui sont pris en charge par un APO spécifique sont définis par l'attribut ListOfManagementServices de l'APO.

6.1.1.2 Spécification du modèle de gestion de la FAL

Les services de gestion spécifiés par le modèle de gestion de la FAL fonctionnent sur des APO de la FAL et leurs attributs. Les classes d'APO de la FAL sont définies à l'intérieur des ASE de la FAL qui sont spécifiés dans le reste des articles de la présente Norme. Chaque classe d'APO de la FAL définie est décrite par un ensemble d'attributs et un ensemble de services.

6.1.1.3 Services du modèle de gestion de la FAL

6.1.1.3.1 Services pris en charge

Les services de gestion sont définis pour gérer les AP et les APO. Pour des raisons de simplicité, AP et APO sont désignés comme des objets dans la suite de ce paragraphe.

Le service Get Attributes est spécifié pour indiquer quand ses attributs peuvent être lus.

6.1.1.3.2 Service Get attributes

6.1.1.3.2.1 Présentation du service

Ce service confirmé est utilisé pour lire les valeurs courantes d'une liste d'attributs pour un ou plusieurs objets d'une ou plusieurs classes définies pour l'AP. Il fonctionne dans un mode "au mieux" (best effort), de telle façon que le service réussit si la valeur pour au moins un attribut demandé pour un objet demandé est retournée

6.1.1.3.2.2 Paramètres de service

Le Tableau 1 présente les paramètres de ce service.

Tableau 1 – Paramètres du service Get attributes

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
List of Objects and their Attributes	S	S (=)		
Key Attribute	M	M (=)		
List of Requested Attributes	M	M (=)		
Result(+)			S	S (=)
Invoke ID				U (=)
More			M	M (=)
List of Responses			M	M (=)
List of Attribute Values			S	S (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante est traitée localement. Voir 1.2.

Argument

L'argument inclut les paramètres de la demande de service.

List of objects and their attributes

Ce paramètre de type de sélection fournit une liste qui identifie chaque objet, sa classe et les attributs qui sont demandées. Trois valeurs pour identifier les attributs sont possibles, tous les attributs, les attributs obligatoires seulement ou une liste d'attributs individuels. Il est présent si l'utilisateur demande des attributs pour une liste d'objets.

Key attribute (Attribut-clé)

Ce paramètre identifie l'objet pour lequel les attributs sont demandés en utilisant l'un des attributs-clés de l'objet.

List of requested attributes

Ce paramètre identifie un ou plusieurs des attributs dont les valeurs sont demandées.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

More

Ce paramètre Boolean (booléen) indique, lorsque sa valeur est FALSE, que les réponses à tous les attributs demandés sont retournées. Si la valeur est TRUE, seules les réponses pour un sous-ensemble d'attributs sont retournées. Cette situation ne se produira que lorsqu'il n'est pas possible que toutes les valeurs tiennent dans une APDU de réponse simple. Lorsque cela se produit, il est nécessaire que l'utilisateur présente une demande supplémentaire pour les attributs manquants. Les valeurs partielles pour les attributs ne sont jamais retournées.

List of responses

Ce paramètre spécifie un indicateur de statut ou une liste de valeurs d'attributs pour chaque objet spécifié dans la demande. Les réponses dans la liste sont retournées dans le même ordre dans lequel les objets ont été spécifiés dans la demande. L'objet associé à chaque réponse n'est explicitement identifié que si la valeur de l'un de ses attributs-clés a été demandée. Si toutes les réponses ne pouvaient pas être retournées dans une simple APDU, le paramètre More serait configuré.

Error status

Ce paramètre est retourné si le Get a échoué pour tous les attributs d'un objet.Il indique la raison la plus probable pour laquelle l'opération a échoué en utilisant la classe d'erreur et le code d'erreur définis pour le service Get Attributes.

List of attribute values

Ce paramètre est retourné si le Get a réussi pour l'un des attributs demandés. Ce paramètre spécifie une liste de valeurs d'attributs, chacune des valeurs étant individuellement identifiée et complète pour l'objet demandé.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.1.1.3.2.3 Procédure de service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Get Attributes est un service "au mieux". "Au mieux" (Best effort) signifie que le service réussit si au moins une valeur est retournée. Si l'utilisateur est en mesure d'obtenir une ou plusieurs des valeurs d'attribut spécifiées, et si celles-ci peuvent être transportées dans une seule APDU, elles sont retournées dans une primitive response (+) de service Get Attributes.

Si elles ne peuvent pas être transportées dans une seule APDU, l'utilisateur retourne les valeurs possibles et il active le fanion More dans la primitive response (+) de service Get Attributes. Le demandeur est alors chargé de présenter des demandes supplémentaires identifiant les attributs qui restent à récupérer.

Si l'utilisateur est incapable d'obtenir la valeur pour au moins un attribut pour un objet dans la liste, le service échoue et l'utilisateur émet une primitive response (-) de service Get Attributes indiquant la raison.

6.1.2 ASE de processus d'application

6.1.2.1 Vue d'ensemble

La présente Norme modélise un Processus d'application (AP). Les Processus d'application représentent les informations et les ressources de traitement d'un système qui peuvent être accessibles par le biais de services de la FAL.

L'élément de service application dans la FAL qui fournit ces services est appelé ASE de processus d'application. Dans l'ASE d'AP, l'AP est modélisé et accessible comme un APO avec un identificateur normalisé et prédéfini.

6.1.2.2 Spécification de la classe AP

6.1.2.2.1 Modèle formel

La classe d'AP spécifie les attributs et les services définis pour des processus d'application. Sa classe mère "top" indique le sommet de l'arbre de la classe de la FAL. L'identificateur numérique pour l'objet AP est toujours 0. L'utilisation de cette valeur réservée permet aux services de gestion d'être utilisés pour les objets AP sans connaître leurs noms.

FAL ASE: **AP ASE**
CLASS: **AP**
CLASS ID: 16
PARENT CLASS: TOP
IDENTIFY, GET STATUS ATTRIBUTES:

1	(m)	Attribute:	List Of AP Service Attributes
1.1	(m)	Attribute:	Manufacturer Identifier
1.1.1	(m)	Attribute:	Vendor Name
1.1.2	(m)	Attribute:	Model Identifier
1.1.3	(m)	Attribute:	Vendor Revision
1.2	(o)	Attribute:	AP Descriptor Reference
1.3	(m)	Attribute:	Network Access State
1.3.1	(m)	Attribute:	Service Access Status
1.3.2	(m)	Attribute:	Operational Status

SYSTEM MANAGEMENT ATTRIBUTES:

2	(m)	Attribute:	List Of System Management Attributes
2.1	(m)	Attribute:	List Of AR Endpoint Info
2.1.1	(m)	Attribute:	Numeric Id
2.1.2	(m)	Attribute:	Configured Max FAL PDU Size Sending
2.1.3	(m)	Attribute:	Configured Max FAL PDU Size Receiving
2.1.4	(m)	Attribute:	Configured Max Outstanding Services Requesting
2.1.5	(m)	Attribute:	Configured Max Outstanding Services Responding
2.1.6	(m)	Attribute:	List of Supported Services
2.2	(m)	Attribute:	List Of In Use AR Endpoint Info
2.2.1	(m)	Attribute:	Context State
2.2.6	(m)	Attribute:	Outstanding Services Requesting Counter
2.2.7	(m)	Attribute:	Outstanding Services Responding Counter

OBJECT MANAGEMENT ATTRIBUTES:

3	(m)	Attribute:	List Of Managed AP Attributes
3.1	(m)	Attribute:	List Of Supported APO Classes and Objects
3.1.1	(m)	Attribute:	List Header
3.1.1.1	(o)	Attribute:	Numeric Identifier = 0
3.1.1.2	(m)	Attribute:	ROM / RAM Flag (TRUE, FALSE)
3.1.1.3	(m)	Attribute:	Max Name Length
3.1.1.4	(m)	Attribute:	Access Protection Supported (TRUE, FALSE)
3.1.1.5	(m)	Attribute:	Version Of List
3.1.1.6	(m)	Attribute:	List Local Reference
3.1.2	(c)	Constraint	Data type Supported
3.1.2.1	(o)	Attribute:	Numeric Identifier = 1
3.1.2.2	(m)	Attribute:	Number Of Data type Objects

3.1.2.3	(m)	Attribute:	Local Reference
3.1.3	(c)	Constraint	Variable
3.1.3.1	(o)	Attribute:	Static Object Numeric Identifier
3.1.3.2	(m)	Attribute:	Number Of Static Objects (Variables)
3.1.3.3	(m)	Attribute:	Static Object Local Reference
3.1.5	(c)	Constraint	Function Invocation Supported
3.1.5.1	(o)	Attribute:	Function Invocation Numeric Identifier
3.1.5.2	(m)	Attribute:	Number Of Function Invocation Objects
3.1.5.3	(m)	Attribute:	Function Invocation Local Reference

SERVICES:

2	(o)	Ops Service:	Identify
3	(o)	Ops Service:	Get Status
5	(o)	Ops Service:	Initiate
6	(o)	Ops Service:	Terminate
8	(o)	Ops Service:	Reject

6.1.2.2.2 Attributs des services Identify, Get Status

Liste des attributs de services d'AP

Les attributs suivants sont accessibles en utilisant les services Identify et Get Status. Sinon, ils ne sont pas accessibles.

Manufacturer identifier

Le Manufacturer Identifier (identificateur du fabricant) est composé des attributs Vendor Name, Model Identifier, Vendor Revision, et facultativement du Serial Number de l'AP. Ces attributs peuvent être lus en utilisant le service Identify.

Vendor name

Cet attribut spécifie le nom du fabricant de l'AP.

Model identifier

Cet attribut spécifie le modèle de l'AP.

Vendor revision

Cet attribut spécifie le niveau de révision de l'AP comme défini par le fabricant.

AP descriptor reference

Cet attribut fait référence à une description générique de l'AP. Le descripteur est un identificateur pour les caractéristiques de l'AP indépendamment de son utilisation. Deux AP, par conséquent, peuvent partager la même description générique. La valeur et l'ensemble autorisé de valeurs pour cet attribut sont définis par l'utilisateur. Cet attribut est utilisé dans le service Initiate.

Network access state

L'attribut Network Access State (état de l'accès au réseau) définit la capacité de l'AP à communiquer. Deux composants sont spécifiés dans la présente norme, Service Access Status et Operational Status. Ces attributs peuvent être lus en utilisant le service Get Status. L'AP peut choisir de les signaler sans être demandé en utilisant le service Status Notification.

Service access status

Cet attribut spécifie les informations relatives à l'état des fonctions de communication de l'AP.

READY FOR COMMUNICATION Tous les services peuvent être utilisés normalement.

LIMITED NUMBER OF SERVICES Un nombre limité de services peut être pris en charge dans certains cas (par exemple pour de petits appareils).

LOADING-NON-INTERACTING Les définitions d'objets sont en cours d'être chargées localement. Par conséquent, le service Begin Set Attributes peut ne pas être utilisé jusqu'à ce que la charge locale soit terminée et que l'état soit modifié.

LOADING-INTERACTING Les définitions d'objets sont en cours d'être chargées sur le réseau (en utilisant le service Set Attributes). Sur l'AR utilisée pour le chargement, seuls les services suivants sont utilisés:

Abort	Get Attributes
Status	Set Attributes
Identify	End Set Attributes

OperationalStatus

Cet attribut donne l'état opérationnel de l'AP réel, comme suit.

OPERATIONAL

PARTIALLY OPERATIONAL

NOT OPERATIONAL

NEEDS MAINTENANCE

6.1.2.2.3 Attributs de la gestion de système

Liste des attributs de gestion système

Les attributs suivants sont accessibles à travers la gestion système. Ils sont utilisés par tous les ASE de l'AP pour renforcer le contrôle de flux des services confirmés (voir les diagrammes d'états dans la CEI 61158-6-8). Sinon, ils ne sont pas accessibles.

List of AR Endpoint Info

Cet attribut spécifie les identificateurs numériques et d'autres informations de configuration (les attributs du context AP) relatives aux AREP associés à l'AP.

Numeric ID

Cet attribut spécifie l'identificateur numérique pour l'AREP.

Configured max FAL PDU size sending

Pour les AR qui ne segmentent pas, cet attribut spécifie le nombre maximal configuré d'octets qui peuvent être envoyés à partir de cet AREP. Sa valeur est égale à la taille maximale de DLSDU moins la taille de celle qui peut être envoyée à partir de cet AREP. Pour les AR qui segmentent, il spécifie le nombre maximal de segments à 256 octets, qui peuvent être envoyés sur cet AREP.

Configured max FAL PDU size receiving

Pour les AR qui ne segmentent pas, cet attribut spécifie le nombre maximal configuré d'octets qui peuvent être reçus sur cet AREP. Sa valeur est égale à la taille maximale de DLSDU moins la taille de celle qui peut être reçue sur cet AREP. Pour les AR qui segmentent, il spécifie le nombre maximal de segments à 256 octets, qui peuvent être reçus sur cet AREP.

Configured max outstanding services requesting (ConfigMaxOsReq)

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés dans le rôle client de cet AREP. Sa valeur est configurée avant l'établissement de l'AR. Cet attribut est utilisé sur les AR client-serveur et poste à poste (c'est-à-dire QUB, bidirectionelles déclenchées par l'utilisateur et placées en file d'attente(Queued user-triggered bi-directional)) et correspond au nombre de diagrammes d'états des transactions d'envoi.

Configured max outstanding services responding (ConfigMaxOsRsp)

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés pour cet AREP comme un serveur. Sa valeur est configurée avant l'établissement de l'AR. Cet attribut est utilisé sur les AR client-serveur et poste à poste (c'est-à-dire QUB) et correspond au nombre de diagrammes d'états des transactions de réception.

List of supported services

Cet attribut spécifie les services que la FAL prend en charge en tant que demandeur et en tant que répondeur. La prise en charge en tant que demandeur indique que la FAL offre à l'AP la possibilité d'initier des primitives "request" de services confirmé et non confirmé et de recevoir des primitives "confirmation" de service confirmé correspondant à partir de la FAL. La prise en charge en tant que répondeur indique que la FAL offre à l'AP la possibilité de délivrer des primitives "indication" de services confirmé et non confirmé à l'AP et de recevoir des primitives "response" de service confirmé à partir de l'AP.

List of in-use AR endpoint info

Cet attribut spécifie les informations dynamiques pour les AREP associés à l'AP, qui participent à une AR-I établie ou qui sont impliqués dans le processus d'établissement de l'AR.

Context state

Cet attribut spécifie l'état de l'AREP tel que vu par l'AP. Les valeurs sont:

CLOSED

OPEN

OPENING-REQUESTING

OPENING-RESPONDING

Outstanding services requesting counter (OsReqCtr)

Cet attribut spécifie le nombre de services confirmés actuellement en cours sur cette AR comme un demandeur.

Outstanding services responding counter (OsRspCtr)

Cet attribut spécifie le nombre de services confirmés actuellement en cours sur cette AR comme un répondeur.

6.1.2.2.4 Attributs de gestion d'objets

List of managed AP attributes

Cet attribut spécifie les attributs de l'AP qui sont accessibles en utilisant les services de l'ASE Management.

List of supported APO classes and objects

Cet attribut spécifie les définitions d'objets maintenues par l'AP.

List header

Cet attribut décrit les caractéristiques des définitions d'objets maintenues par l'AP.

Numeric ID

Cet attribut est l'ID numérique de l'attribut "List Header" (En-tête de Liste). Sa valeur est toujours 0.

ROM / RAM flag

Cet attribut, lorsque sa valeur est TRUE, indique que les modifications apportées aux définitions d'objets sont autorisées.

Max name length

Cet attribut spécifie la longueur maximale en octets pour les noms d'objets définis pour cet AP.

Access protection supported

Cet attribut, lorsque sa valeur est TRUE, indique que la protection d'accès est prise en charge.

Version of list

Cet attribut indique la version courante de cette liste. Chaque mise à jour de la liste de définitions d'objets, qu'elle soit localement initiée ou apportée par l'utilisation du service SetAttributes, provoque l'incrémentation du numéro de version.

List local reference

Cet attribut fait référence à la liste qui a une signification locale, telle qu'une adresse. La valeur hexadécimale FFFFFFFF indique qu'aucune référence locale n'est disponible.

XYZ object class supported constraint (contrainte prise en charge par la classe d'objets spécifiée par XYZ)

Cette contrainte choisit les classes d'objets spécifiées par XYZ. Les attributs qui suivent la contrainte s'appliquent aux définitions d'objets de la classe sélectionnée d'objets.

XYZ numeric ID

Cet attribut est l'ID numérique du premier objet dans une série d'objets de classe(s) choisie(s) par la contrainte. Tous les objets de la classe ou des classes sélectionnée(s) sont contenus dans la série. Le premier Numeric ID (identificateur numérique) pour les types Data est toujours 1.

Number of XYZ entries

Cet attribut spécifie le nombre d'objets dans la série.

XYZ local reference

Cet attribut fait référence au début de définitions d'objets décrivant les objets dans la série. La référence a une signification locale, comme une adresse. La valeur hexadécimale FFFFFFFF indique qu'aucune référence locale n'est disponible.

6.1.2.2.5 Services

Identify

Ce service facultatif est utilisé pour demander des informations relatives au fabricant à partir de l'AP.

Get Status

Ce service facultatif est utilisé pour demander à l'AP l'état network-visible ("visible par le réseau").

Initiate

Ce service facultatif est utilisé pour ouvrir le contexte AP pour une relation d'applications.

Terminate

Ce service facultatif est utilisé pour fermer brusquement le contexte AP pour une relation d'applications.

Reject

Ce service est initié en interne par l'ASE de l'AP pour indiquer qu'une APDU de demande de service confirmé a été reçue avec une erreur de protocole.

6.1.2.3 Spécification de service de l'ASE "Application process"

6.1.2.3.1 Services pris en charge

Ce paragraphe contient la définition de services qui sont propres à cet ASE. Les services définis pour cet ASE sont:

Identify

Get Status

Initiate

Terminate

Reject

6.1.2.3.2 Service Identify

6.1.2.3.2.1 Présentation du service

Les informations permettant d'identifier un AP sont lues avec ce service.

NOTE L'attribut ProfileNumber de l'AP est émis avec le service Initiate.

6.1.2.3.2.2 Primitives de service

Le Tableau 2 présente les paramètres de ce service.

Tableau 2 – Identify

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Result(+)			S	S (=)
Invoke ID				U (=)
Vendor Name			M	M (=)
Model Identifier			M	M (=)
Vendor Revision			M	M (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante est traitée localement. Voir 1.2.

Argument

L'argument inclut les paramètres de la demande de service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Vendor name

Ce paramètre spécifie la valeur de l'attribut Vendor Name (nom du vendeur) de l'AP.

Model identifier

Ce paramètre spécifie la valeur de l'attribut Model Identifier (identificateur du modèle) de l'AP.

Vendor revision

Ce paramètre spécifie la valeur de l'attribut Vendor Revision (révision vendeur) de l'AP.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.1.2.3.2.3 Procédure de service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

6.1.2.3.3 Service Get status

6.1.2.3.3.1 Présentation du service

L'état de l'appareil/de l'utilisateur est lu avec le service Get Status.

6.1.2.3.3.2 Primitives de service

Le Tableau 3 présente les paramètres de ce service.

Tableau 3 – Get status

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Invoke ID	U			
Result(+)			S	S (=)
Invoke ID				U (=)
Service Access Status			M	M (=)
Operational Status			M	M (=)
Local Detail			U	U (=)
Result(-)			S	S (=)
Invoke ID				U (=)
Error Info			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante est traitée localement. Voir 1.2.

Argument

L'argument inclut les paramètres de la demande de service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service access status

Ce paramètre spécifie la valeur de l'attribut Service Access Status (statut de l'accès au service) de l'AP.

Operational status

Ce paramètre spécifie la valeur de l'attribut Operational Status (statut opérationnel) de l'AP.

Local detail

Ce paramètre spécifie l'état défini par l'utilisateur de l'application.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.1.2.3.3.3 Procédure de service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

6.1.2.3.4 Service Initiate

6.1.2.3.4.1 Présentation du service

Ce service est utilisé pour établir un contexte entre les AP communicants pour l'information sur les échanges sur une simple AR. Le service Initiate négocie les services de la FAL pris en charge, les services en cours maximaux autorisés, la longueur maximale de PDU et la version courante des définitions d'objets. Les services de la FAL pris en charge, les services en cours

maximaux autorisés, la longueur maximale de PDU sont des attributs de gestion système de l'AP, configurés pour l'AREP.

6.1.2.3.4.2 Primitives de service

Le Tableau 4 présente les paramètres de ce service.

Tableau 4 – Initiate

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
Version Object Definitions Calling	M	M (=)		
AP Descriptor Calling	M	M (=)		
Access Protection Supported Calling	M	M (=)		
Password Calling	M	M (=)		
Access Groups Calling	M	M (=)		
Result(+)			S	S (=)
Version OD Called			M	M (=)
AP Descriptor Called			M	M (=)
Access Protection Supported Called			M	M (=)
Password Called			M	M (=)
Access Groups Called			M	M (=)
Result(-)			S	S (=)
Error Code			M	M (=)
Max PDU Length Sending Called				C
Max PDU Length Receiving Called				C
FAL Features Supported Called				C

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante est traitée localement. Voir 1.2.

Argument

Ce paramètre contient les paramètres de l'invocation de service.

Version object definitions calling

Ce paramètre spécifie la version des définitions d'objets du client. Sa valeur est nulle si le client ne contient pas de définitions d'objet.

AP descriptor calling

Ce paramètre spécifie la valeur de l'attribut Descriptor Reference (référence du descripteur) de l'AP appelant s'il en possède. S'il n'existe pas, sa valeur est nulle.

Access protection supported calling

Ce paramètre spécifie la valeur de l'attribut Access Protection Supported (protection d'accès prise en charge) de l'AP appelant s'il en possède. S'il n'existe pas, sa valeur est nulle.

Password calling

Ce paramètre spécifie le mot de passe à utiliser pour l'accès à tous les objets du serveur via cette AR. Si l'accès par mot de passe ne doit pas être utilisé sur cette AR, sa valeur est nulle.

Access groups calling

Ce paramètre spécifie l'appartenance du client dans des groupes d'accès spécifiques. L'appartenance s'applique pour l'accès à tous les objets du serveur via cette AR.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Version object definition called

Ce paramètre spécifie la version des définitions d'objets du serveur.

AP descriptor called

Ce paramètre spécifie la valeur de l'attribut Descriptor Reference (référence du descripteur) de l'AP appelé.

Access protection supported called

Ce paramètre spécifie la valeur de l'attribut Access Protection Supported (protection d'accès prise en charge) de l'AP appelé s'il en possède. S'il n'existe pas, sa valeur est nulle.

Password called

Ce paramètre spécifie le mot de passe à utiliser pour l'accès à tous les objets du client via cette AR. Si l'accès par mot de passe ne doit pas être utilisé sur cette AR, sa valeur est nulle.

Access groups called

Ce paramètre spécifie l'appartenance du serveur dans des groupes d'accès spécifiques. L'appartenance s'applique pour l'accès à tous les objets du client via cette AR.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Error code

Ce paramètre fournit la cause de l'échec.

Raison	Signification
Max FAL PDU Size Insufficient	La longueur maximale de PDU n'est pas suffisante pour la communication.
Service Not Supported	Le service ou l'option demandé(e) n'est pas pris(e) en charge par le serveur.
User Initiate Denied	L'utilisateur de la FAL refuse d'établir la connexion.
Version Object Definition incompatible	Les versions appelé et appelant des définitions d'objets ne sont pas compatibles.
Password Error	Il existe déjà une AR établie avec le même mot de passe ou le mot de passe n'est pas valide.
AP Descriptor Incompatible	Le descripteur n'est pas pris en charge par le serveur.
Other	Autre raison différente de l'une des raisons identifiées ci-dessus.

Max PDU length sending called

Ce paramètre spécifie la longueur maximale de la FAL PDU à envoyer et qui peut être traitée sur cette AR.

Il est émis par la FAL appelée et fait partie de la primitive Initiate.cnf.

Max PDU length receiving called

Ce paramètre spécifie la longueur maximale de la FAL PDU à recevoir et qui peut être traitée sur cette AR.

Si pris en charge, il est émis par la FAL appelée et fait partie de la primitive Initiate.cnf.

FAL services supported called

Ce paramètre identifie les services AL facultatifs et les options prises en charge par le serveur (voir AR List).

Si pris en charge, il est émis par la FAL appelée et fait partie de la primitive Initiate.cnf.

6.1.2.3.4.3 Procédure de service

Ce service fonctionne à travers une file d'attente.

L'utilisateur demandeur présente la primitive "request" de service à son entité d'application de la FAL et démarre un temporisateur associé pour surveiller la demande. L'ASE de l'AP construit le corps de l'APDU "Initiate Request" et l'achemine sur l'AR spécifiée en utilisant la primitive "request" de service AR Establish. Il crée également un diagramme d'états de transactions.

Dès réception du corps de l'APDU Initiate Request dans une primitive "indication" de service AR Establish, l'ASE de l'AP répondeur le décode. Si une erreur de protocole s'est produite lors du décodage du corps de l'APDU reçue, l'ASE utilise le service AR Abort pour arrêter prématurément l'AR. Si une erreur de protocole ne s'est pas produite, l'ASE délivre à son utilisateur primitive "indication" de service Initiate.

Si l'utilisateur répondeur est en mesure de traiter la demande avec succès, l'utilisateur retourne une primitive response (+) de service Initiate.

Si l'utilisateur répondeur est incapable de traiter la demande avec succès, le service échoue et l'utilisateur émet une primitive response (-) de service Initiate indiquant la raison.

L'ASE de l'AP répondeur construit un corps de l'APDU de réponse de service Initiate pour une primitive response (+) ou un corps de l'APDU d'erreur de service Initiate pour une primitive response (-) et l'achemine sur l'AR spécifiée à l'aide d'une primitive "response" de service AR Establish.

Dès réception du corps de l'APDU retourné dans une primitive confirmation (+ ou -) de service AR Establish, l'ASE émetteur envoie une primitive "confirmation" de service Initiate à son utilisateur qui spécifie s'il s'agit d'un succès ou d'un échec, et la raison de l'échec en cas d'échec. Il annule également le diagramme d'états de transactions correspondant.

Toutefois, si le temporisateur de transaction de l'AP expire avant la réception de l'APDU de réponse ou d'erreur correspondante, l'AP peut prendre des mesures correctives, ce qui peut inclure l'instruction à son ASE local d'annuler le diagramme d'états de transactions. Une telle instruction à l'ASE pourrait se produire à travers une interface localement définie.

6.1.2.3.5 Service Terminate

6.1.2.3.5.1 Présentation du service

Ce service est utilisé pour libérer une AR existante entre les AP ou pour terminer l'implication d'un AREP abonné/récepteur dans une AR.

6.1.2.3.5.2 Primitives de service

Le Tableau 5 présente les paramètres de ce service.

Tableau 5 – Terminate

Nom de paramètre	Req	Ind
Argument		
AREP	M	M
Locally Generated		M
Terminate Identifier	M	M (=)
Reason Code	M	M (=)
Terminate Detail	U	U (=)

Argument

Ce paramètre contient les paramètres de l'invocation de service.

Locally generated

Ce paramètre indique si la terminaison était générée localement ou par le partenaire de communication.

La valeur False n'est pas autorisée si le paramètre Terminate Identifier a la valeur FAL et le paramètre Reason Code a la valeur AR Error.

Terminate identifier

Ce paramètre indique où a été détectée la raison pour la terminaison. Les valeurs possibles sont:

FAL USER
FAL APO ASE
FAL AR ASE
DLL

Reason code

Ce paramètre spécifie la raison pour la terminaison.

Si le paramètre Terminate Identifier a la valeur USER, les valeurs suivantes s'appliquent.

	Signification
Disconnection	La connexion est libérée par l'utilisateur de la FAL.
Version Object Definition incompatible	Les versions appelées et appelant des définitions d'objets ne sont pas compatibles.
Password Error	Il existe déjà une AR établie avec le même mot de passe ou le mot de passe n'est pas valide.
Descriptor incompatible	Le descripteur du serveur n'est pas pris en charge par le client.
Limited Services Permitted	L'AP est da statut logique LIMITED-SERVICES-PERMITTED.

Si le paramètre Abort Identifier a la valeur FAL, les valeurs suivantes s'appliquent:

Raison	Signification
AR Error	AR erronée
User Error	Primitive de service incorrecte, inconnue ou défectueuse reçue à partir de l'utilisateur de la FAL
FAL PDU Error	PDU de la FAL inconnue ou défectueuse reçue à partir de l'ASE de l'AR
Connection State Conflict AR ASE	Primitive de service de l'ASE de l'AR incorrecte
AR ASE Error	Primitive de service de l'ASE de l'AR inconnue ou défectueuse
PDU Size	La longueur de la PDU dépasse la longueur maximale de la PDU
Feature Not Supported	La PDU SERVICE-REQ est reçue à partir de l'ASE de l'AR alors que le service ou l'option n'est pas pris(e) en charge comme un serveur (se référer à l'attribut FAL Features Supported dans l'AR)
Invoke ID Error Response	Le service confirmé service.rsp est reçu à partir de l'utilisateur de la FAL et Invoke ID n'existe pas ou CONFIRMED_SERVICE-RSP_PDU est reçue à partir de l'ASE de l'AR et Invoke ID n'existe pas
Max Services Overflow	La CONFIRMED_SERVICE-REQ_PDU est reçue à partir de l'ASE de l'AR et Outstanding Services Counter Receiving ≥ Max Outstanding Services Receiving (c'est-à-dire que le compteur des services en cours en réception est supérieur ou égal à la valeur maximale des services en cours en réception)
Connection State Conflict	La INITIATE-REQ_PDU est reçue à partir de l'ASE de l'AR
Service Error	Le service dans la primitive "response" ne correspond pas au service dans la primitive "indication" ou le service dans la primitive "confirmation" ne correspond pas au service dans la primitive "request"
Invoke ID Error Request	La CONFIRMED_SERVICE-REQ_PDU reçue à partir de l'ASE de l'AR et Invoke ID existe déjà.

Si le paramètre Terminate Identifier a une valeur AR ASE ou DLL, le paramètre Reason Code est fourni par l'ASE de l'AR.

Terminate detail

Ce paramètre facultatif spécifie des informations supplémentaires concernant la raison de l'arrêt brusque (maximum 16 octets). Dans le cas de rapports d'erreurs à partir de l'application, la signification est définie dans le profil.

6.1.2.3.5.3 Procédure de service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

6.1.2.3.6 Service Reject

6.1.2.3.6.1 Présentation du service

Ce service est utilisé pour informer le point d'extrémité distant qu'une erreur de protocole a été détectée. Il est généré par l'ASE de l'AP si une APDU a été reçue avec un code de type de service non valide. Lorsqu'un autre ASE d'APO détecte une erreur de protocole, il demande en interne à l'ASE de l'AP de générer une PDU Reject.

6.1.2.3.6.2 Primitives de service

Le Tableau 6 présente les paramètres de ce service.

Tableau 6 – Reject

Nom de paramètre	Req	Ind
Argument		
AREP	M	M
Reject Code	M	M (=)
Original FAL Service Type	M	M (=)
Original Invoke ID	M	M (=)
Original PDU Body	U	U (=)

Argument

L'argument inclut les paramètres de la demande de service.

Reject code

Ce paramètre indique la raison pour le rejet. Les valeurs de cette raison de rejet peuvent indiquer:

Unsupported Service

AR Not Established

AR Not Defined

Max Outstanding Requests Exceeded

Max PDU Size Exceeded

Duplicate Invoke ID

Original FAL service type

Ce paramètre spécifie le type de service de la demande du service rejeté.

Original invoke ID

Ce paramètre spécifie l'Invoke ID de la demande du service rejeté.

Original PDU body

Ce paramètre facultatif spécifie une partie ou l'intégralité des données de l'APDU qui était rejetée. La quantité de données incluse est déterminée par l'utilisateur.

6.1.2.3.6.3 Procédure de service

Le service Reject est un service qui fonctionne à travers une file d'attente ou une mémoire tampon. Il ne peut être initié que par l'ASE de l'AP de la FAL.

Si un ASE d'APO reçoit une APDU de demande de service confirmé qui contient une erreur de protocole, l'ASE de l'AP construit une APDU Reject Request et la renvoie sur l'AR spécifiée.

Dès réception de l'APDU Reject Request, l'ASE de l'AP destinataire délivre une primitive AR-Reject.indication à travers l'ASE de la FAL qui a émis la primitive "request" du service d'envoi confirmé, et il émet une confirmation négative à l'utilisateur demandeur.

6.1.3 ASE de relation d'applications

6.1.3.1 Vue d'ensemble

6.1.3.1.1 Généralités

Dans un système distribué, les processus d'application communiquent les uns avec les autres en échangeant des messages de couche Application à travers des voies de communication bien définies de la couche Application. Ces voies de communication sont modélisées dans la couche FAL en relations d'applications (AR, application relationship).

Les AR sont chargées d'acheminer des messages entre des applications en fonction des caractéristiques de communications spécifiques requises par les systèmes à temps critique. Des combinaisons différentes de ces caractéristiques conduisent à la définition de types d'AR