IEC 61158-5-4

Edition 1.0 2007-12

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 5-4: Application layer service definition – Type 4 elements

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE XB

ICS 25.040.40; 35.100.70

ISBN 2-8318-9446-8

CONTENTS

FOREWORD 4
INTRODUCTION 5
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 10
3.6 Fieldbus application layer specific definitions 10
3.7 Abbreviations and symbols 15
3.8 Conventions 17
4 Concepts 20
4.1 Overview 20
4.2 Architectural relationships 20
4.3 Fieldbus Application Layer structure 22
4.4 Fieldbus Application Layer naming and addressing 34
4.5 Architecture summary 35
4.6 FAL service procedures 36
4.7 Common FAL attributes 37
4.8 Common FAL service parameters 37
4.9 APDU size 38
5 Type 4 communication model specification 38
5.1 Concepts 38
5.2 Variable ASE 45
5.3 Application relationship ASE 64
Bibliography 71

Figure 1 – Relationship to the OSI basic reference model 20
Figure 2 – Architectural positioning of the fieldbus Application Layer 21
Figure 3 – Client/server interactions 24
Figure 4 – Pull model interactions 25
Figure 5 – Push model interactions 25
Figure 6 – APOs services conveyed by the FAL 27
Figure 7 – Application entity structure 29
Figure 8 – Example FAL ASEs 30
Figure 9 – FAL management of objects 31
Figure 10 – ASE service conveyance 32
Figure 11 – Defined and established AREPs 34
Figure 12 – FAL architectural components 36

Figure 13 – FAL AE 39
Figure 14 – Summary of the FAL architecture 42
Figure 15 – FAL service procedure overview 43
Figure 16 – Time sequence diagram for the confirmed services 44
Figure 17 – Time sequence diagram for unconfirmed services 45

Table 1 – REQUEST service parameters 60
Table 2 – RESPONSE service parameters 61
Table 3 – Error codes by source 62
Table 4 – Reserve REP service parameters 62
Table 5 – Free AREP service parameters 63
Table 6 – Get REP attribute service parameters 63
Table 7 – Set REP attribute service parameters 64
Table 8 – AR send service parameters 68
Table 9 – AR acknowledge service parameters 68
Table 10 – AR get attributes service parameters 69
Table 11 – AR set attributes service parameters 69

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-4: Application layer service definition – Type 4 elements

FOREWORD

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

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

International Standard IEC 61158-5-4 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 part and its Type 4 companion parts also cancel and replace IEC/PAS 62412, published in 2005..

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

a) deletion of the former Type 6 fieldbus for lack of market relevance;

b) addition of new types of fieldbuses;

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

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

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

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

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 5-4: Application layer service definition – Type 4 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 4 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 4 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 4 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 2 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*

IEC 61158-3-4, *Industrial communication networks – Fieldbus specifications – Part 3-4: Data-link layer service definition – Type 4 elements*

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

IEC 61158-6-4, *Industrial communication networks – Fieldbus specifications – Part 6-4: Application layer protocol specification – Type 4 elements*

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 – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

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

ISO/IEC 10646-1, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Architecture and Basic Multilingual Plane*

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 as defined in these publications apply:

3.1 ISO/IEC 7498-1 terms

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

3.2 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 apply.

a) DL-Time
b) DL-Scheduling-policy
c) DLCEP
d) DLC
e) DLPDU
f) DLSDU
g) DLSAP
h) fixed tag
i) generic tag
j) link
k) network address
l) node address
m) node
n) tag
o) scheduled
p) unscheduled

3.6 Fieldbus application layer specific definitions

For the purposes of this standard, the following terms and definitions apply.

3.6.1
application
function or data structure for which data is consumed or produced

3.6.2
application objects
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

3.6.3
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.6.4
application process identifier
distinguishes multiple application processes used in a device

3.6.5
application process object
component of an application process that is identifiable and accessible through an FAL application relationship

NOTE Application process object definitions are composed of a set of values for the attributes of their class (see the definition for Application Process Object Class 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.

3.6.6
application process object class
a class of application process objects defined in terms of the set of their network-accessible attributes and services

3.6.7
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation. This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities

3.6.8
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.6.9
application relationship endpoint
context and behavior of an application relationship as seen and maintained by one of the application processes involved in the application relationship

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

3.6.10
attribute
description of an externally visible characteristic or feature of an object

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

3.6.11
behaviour
indication of how an object responds to particular eventss

3.6.12
bit-no
designates the number of a bit in a bitstring or an octet

3.6.13
channel
single physical or logical link of an input or output application object of a server to the process

3.6.14
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.

3.6.15
class attributes
attribute that is shared by all objects within the same class

3.6.16
class code
unique identifier assigned to each object class

3.6.17
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.

3.6.18
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

3.6.19
communication objects
components that manage and provide a run time exchange of messages across the network

EXAMPLES: Connection Manager object, Unconnected Message Manager (UCMM) object, and Message Router object

3.6.20
connection
logical binding between application objects that may be within the same or different devices

NOTE 1 Connections may be either point-to-point or multipoint.

3.6.21
conveyance path
unidirectional flow of APDUs across an application relationship

3.6.22
dedicated AR
AR used directly by the FAL User

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

3.6.23
default DL-address
value 126 as an initial value for DL-address, which has to be changed (e.g. by assignment of an DL-address via the fieldbus) before operation with a DP-master (class 1)

3.6.24
device
physical hardware connected to the link

NOTE A device may contain more than one node.

3.6.25
dynamic AR
AR that requires the use of the AR establishment procedures to place it into an established state

3.6.26
endpoint
one of the communicating entities involved in a connection

3.6.27
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.6.28
error class
general grouping for related error definitions and corresponding error codes

3.6.29
error code
identification of a specific type of error within an error class

3.6.30
event
an instance of a change of conditions

3.6.31
FAL subnet
subnetworks composed of one or more data link segments, identified by a subset of the network address

NOTE FAL subnets are permitted to contain bridges but not routers.

3.6.32
FIFO variable
a Variable Object class, composed of a set of homogeneously typed elements, where the first written element is the first element that can be read

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

3.6.33
frame
denigrated synonym for DLPDU

3.6.34
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

3.6.35
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. Also for service invocations, an Invoke ID may be used to unambiguously identify the service invocation and differentiate it from other outstanding service invocations.

3.6.36
index
address of an object within an application process

3.6.37
instance
the actual physical occurrence of an object within a class that identifies one of many objects within the same object class

EXAMPLE California is an instance of the object class state.

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

3.6.38
instance attributes
attribute that is unique to an object instance and not shared by the object class

3.6.39
instantiated
object that has been created in a device

3.6.40
logical device
a certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation 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
module
hardware or logical component of a physical device

3.6.46
network
a set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.6.47
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

3.6.48
object specific service
service unique to the object class which defines it

3.6.49
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.6.50
physical device
an automation or other network device

3.6.51
property
a general term for descriptive information about an object

3.6.52
provider
source of a data connection

3.6.53
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.

3.6.54
publishing manager
role of an AR endpoint in which it issues one or more confirmed service request APDUs to a publisher to request the publisher to publish a specified object. Two types of publishing managers are defined by this standard, pull publishing managers and push publishing managers, each of which is defined separately

3.6.55
pull subscriber
type of subscriber that recognizes received confirmed service response APDUs as published object data

3.6.56
resource
a processing or information capability of a subsystem

3.6.57
route endpoint
object container containing Variable Objects of a variable class

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 End Point
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Cnf	Confirmation
CR	Communication Relationship
CREP	Communication Relationship End Point
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLCEP	Data Link Connection End Point
DLL	Data Link Layer
DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	DL-service-data-unit
DNS	Domain Name Service
DP	Decentralised Peripherals
FAL	Fieldbus Application Layer
FIFO	First In First Out
HMI	Human-Machine Interface
ID	Identifier
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
Ind	Indication
IP	Internet Protocol
ISO	International Organization for Standardization
LDev	Logical Device
LME	Layer Management Entity
OSI	Open Systems Interconnect
PDev	Physical Device
PDU	Protocol Data Unit
PL	Physical Layer
QoS	Quality of Service
REP	Route Endpoint
Req	Request
Rsp	Response
RT	Runtime
SAP	Service Access Point
SCL	Security Level
SDU	Service Data Unit
SEM	State event matrix

SMIB System Management Information Base

SMK System Management Kernel

STD State transition diagram, used to describe object behaviour

VAO Variable Object

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

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

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

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

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

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.

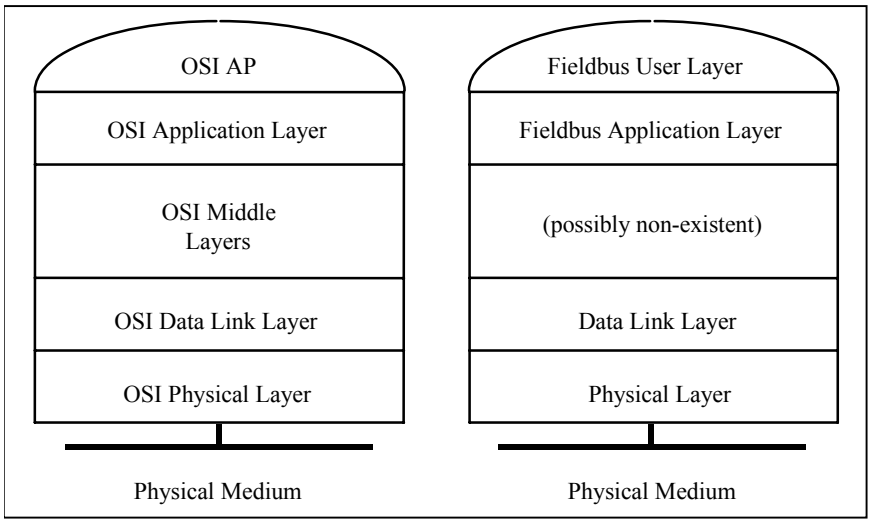

Figure 1 – Relationship to the OSI basic reference model

4.2.2 Relationships to other fieldbus entities

4.2.2.1 General

The 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 fieldbus 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.

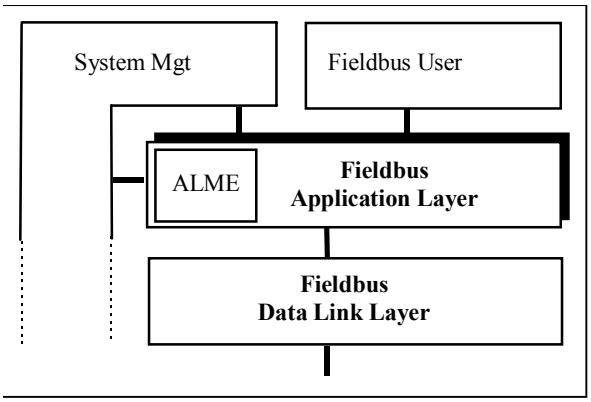

Figure 2 – Architectural positioning of the fieldbus Application Layer

4.2.2.2 Use of the fieldbus 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).

4.2.2.3 Support for fieldbus applications

Fieldbus 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.

4.2.2.4 Support for system management

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

4.2.2.5 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 fieldbus 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 (IEC 61158-3 and IEC 61158-4) 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. IEC 61158-6 specifies the protocols necessary to convey these object services between applications.

4.3.3 Fieldbus application processes

4.3.3.1 Definition of the fieldbus AP

In the fieldbus environment, an application may be partitioned into a set of components and distributed across a number of devices on the network. Each of these components is referred to as a fieldbus 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

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

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

For example, in the fieldbus environment, interactions may be based on request/response messages sent directly between APs, or on data/events sent by one AP for use by others. These two models of 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.

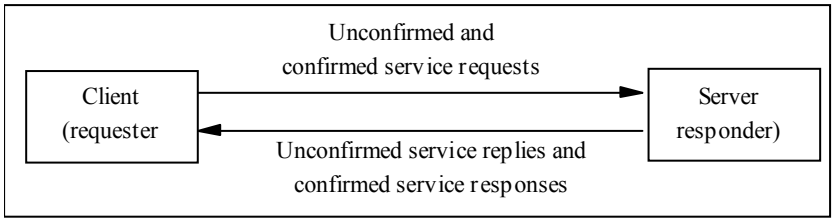

Figure 3 – Client/server interactions

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

4.3.3.3.3.2 Pull model interactions

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

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

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

The second difference occurs in the behavior of the subscribers. Pull model subscribers, referred to as pull subscribers, are capable of accepting published data in confirmed service responses without having issued the corresponding request. Figure 4 illustrates these concepts.

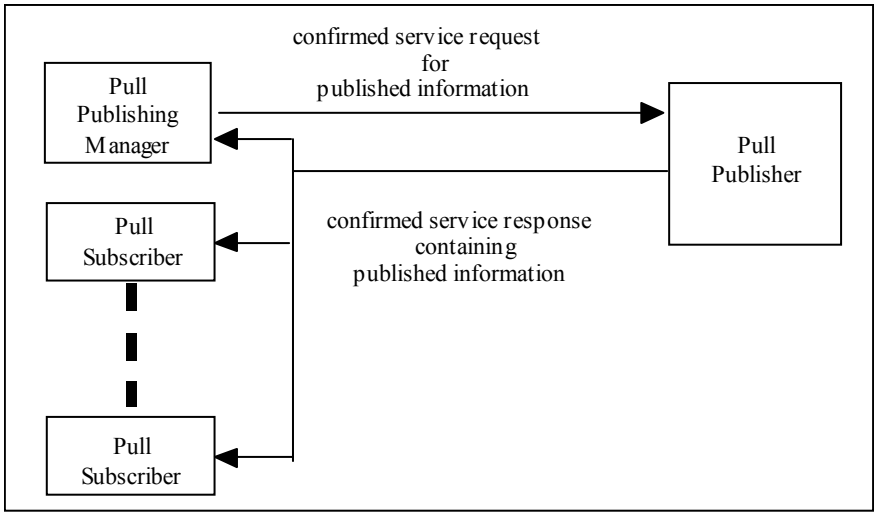

Figure 4 – Pull model interactions

4.3.3.3.3.3 Push model interactions

In the "push" model, two services may be used, one confirmed and one unconfirmed. The confirmed service is used by the subscriber 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 5 illustrates the concept of the Push Model.

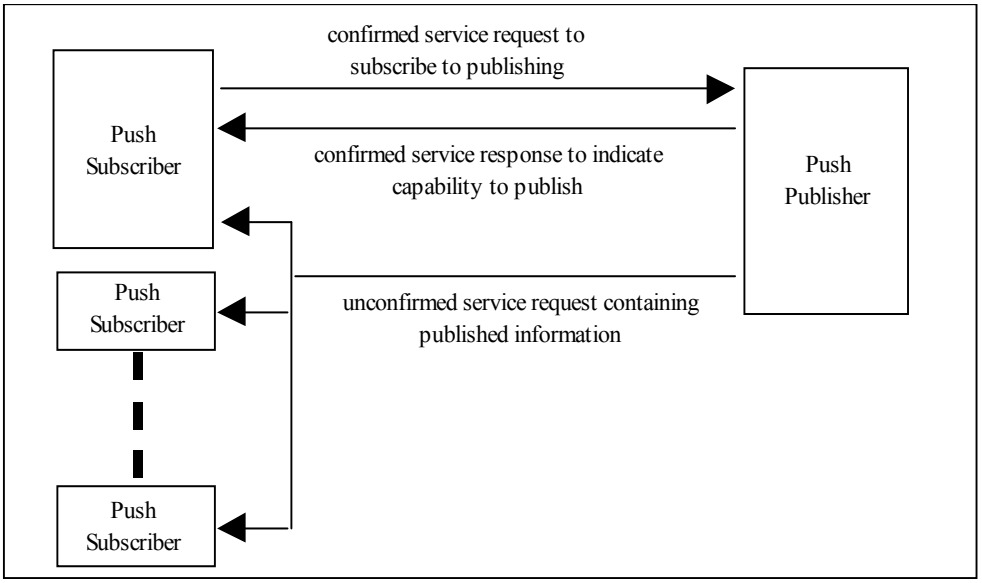

Figure 5 – Push model interactions

4.3.3.3.3.4 Timeliness of published information

To support the perishable nature of published information, the FAL may support four types of timeliness defined for publisher/subscriber interactions. Each make it possible for subscribers of published data to determine if the data they are receiving is up-to-date or "stale". These types are realized through mechanisms within the Data Link Layer (DLL). Each is described briefly below. For a more detailed description, refer to IEC 61158-3 and IEC 61158-4.

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

4.3.3.4 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 fieldbus AP, there is one and only one FAL AE. APOs are the network representation of application specific capabilities (user application process objects) of an AP that are accessible through its FAL AE.

4.3.3.5 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 6, are provided by the FAL communications entity of the AP, known as the FAL Applications Entity (FAL AE).

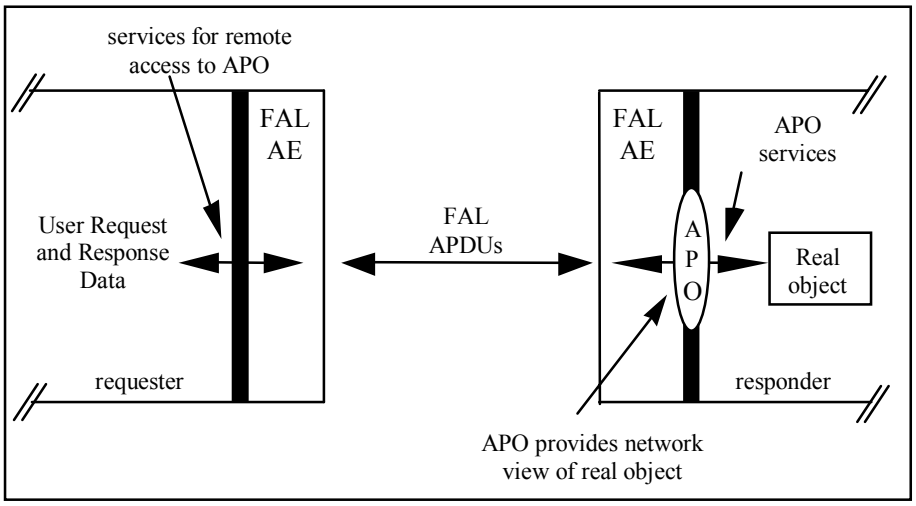

Figure 6 – APOs services conveyed by the FAL

In Figure 6, remote APs acting as clients may access the real object by sending requests through the APO that represents the real object. Local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

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

4.3.4.2 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 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 7 illustrates this concept.

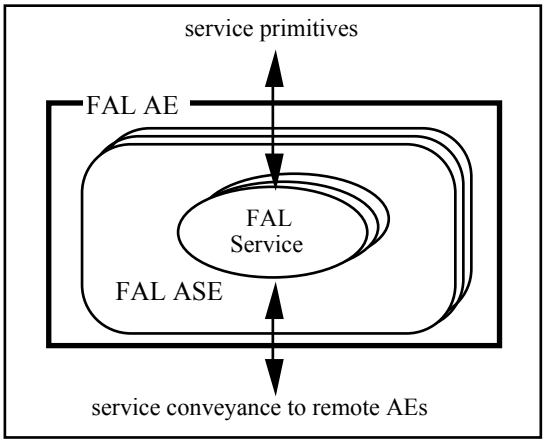

Figure 7 – Application entity structure

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 Fieldbus 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 is used by the requesting AP and its FAL AE to associate the response with the appropriate request.

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

4.3.6.3 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 8 illustrates an example set of the FAL ASEs and their architectural relationships. All APO ASEs follow this example.

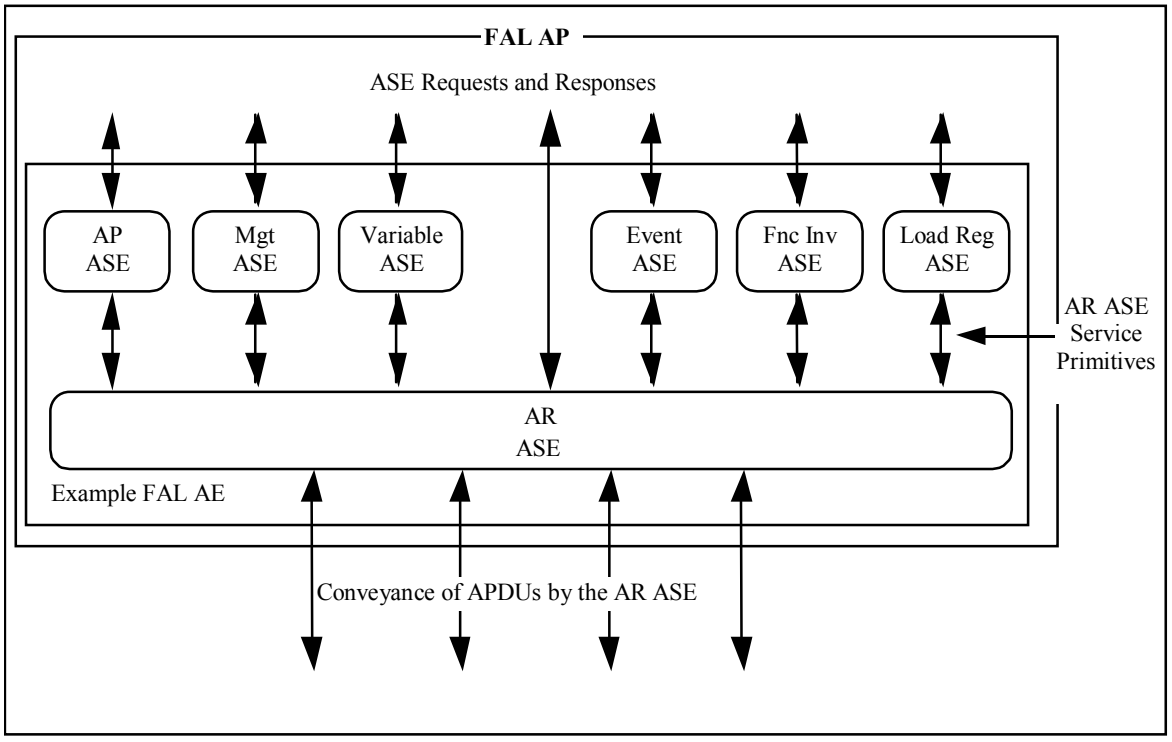

Figure 8 – Example FAL ASEs

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 9 illustrates the integration of management and operational services for an object within an AP.

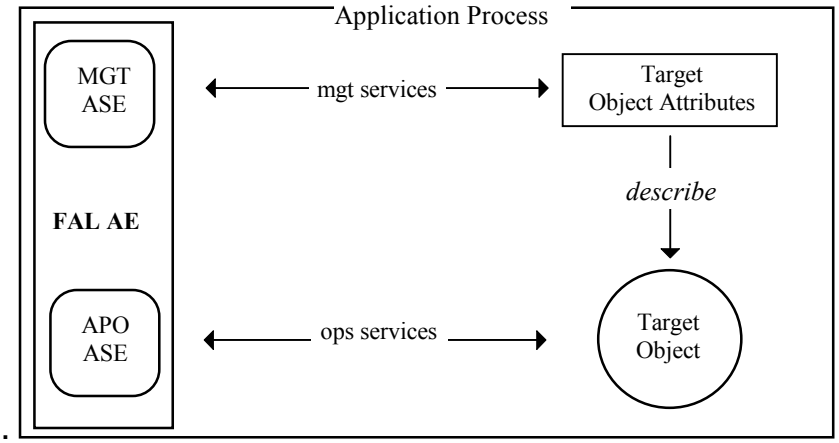

Figure 9 – FAL management of objects

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 10 illustrates these concepts.

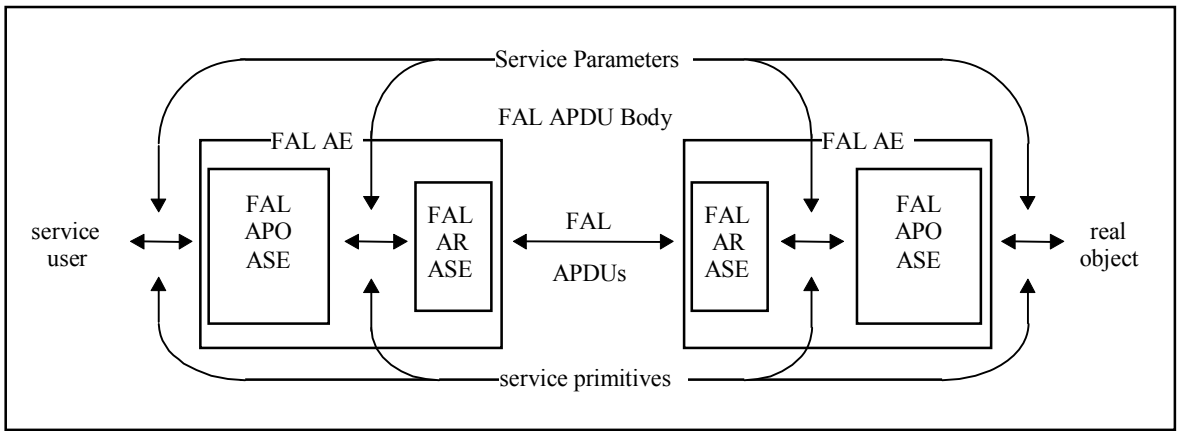

Figure 10 – ASE service conveyance

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. Those that convey services from one AP to a number of APs have a cardinality of 1-to-many. Those that convey services from/to multiple APs have a cardinality of many-to-many.

4.3.7.5 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.

4.3.7.6 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.

4.3.7.7 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.

4.3.7.8 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.

4.3.7.9 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.

4.3.7.10 AREP timeliness

AREPs convey APDUs between applications using the services of the Data Link Layer. When the timeliness capabilities are defined for an AREP and supported by the Data Link Layer, the AREP forwards the timeliness indicators provided by the Data Link Layer. These timeliness

indicators make it possible for subscribers of published data to determine if the data they are receiving is up-to-date or “stale”.

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

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

4.3.7.11 Definition and creation of AREPs

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

AREPs may be pre-defined and pre-established, or they may be pre-defined and dynamically established. Figure 11 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).

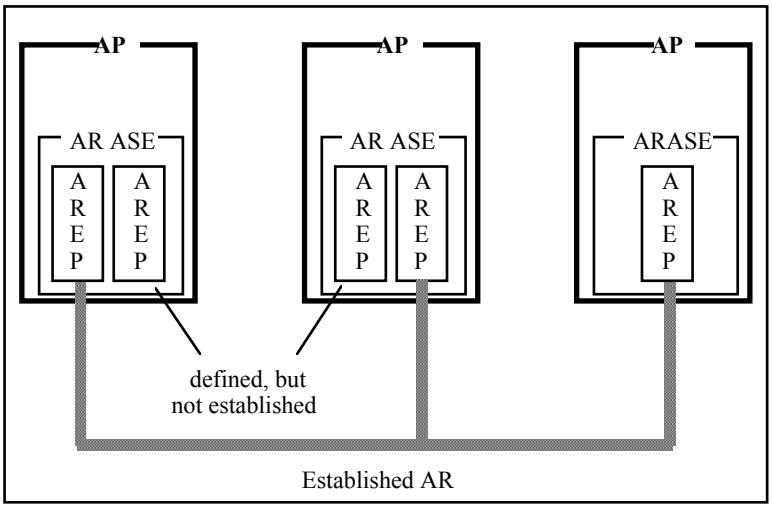

Figure 11 – Defined and established AREPs

4.3.7.12 AR establishment and termination

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

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

4.4 Fieldbus Application Layer 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 12 illustrates the major components of the FAL architecture and how they relate to each other.

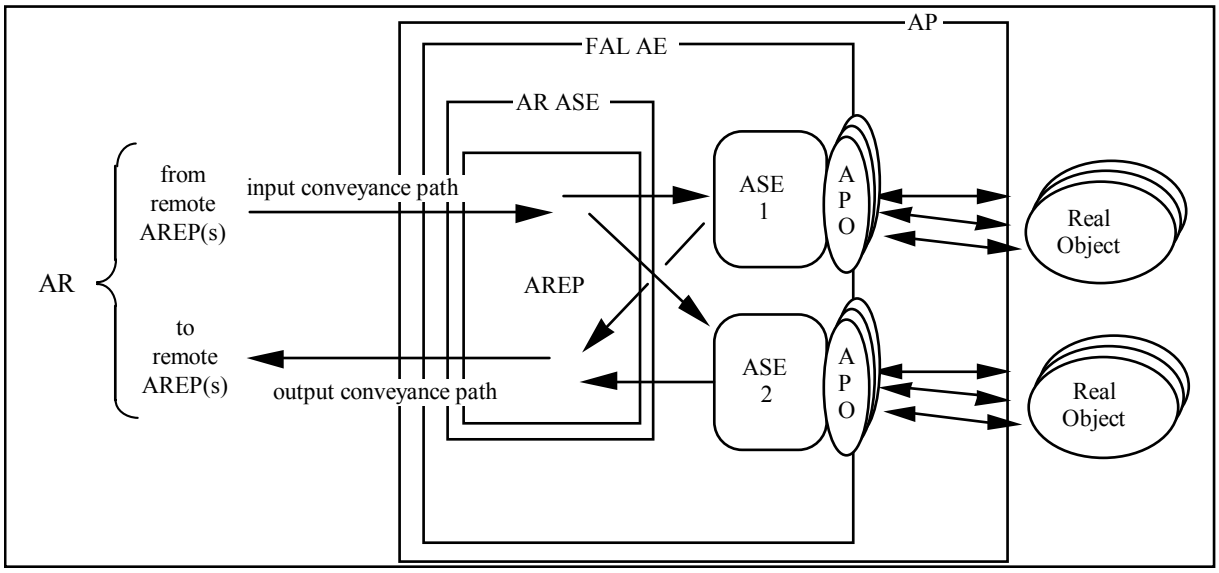

Figure 12 – FAL architectural components

Figure 12 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

The requesting user submits a confirmed service request primitive to its FAL. The appropriate FAL ASE builds the related confirmed service request APDU Body and conveys it on the specified AR.

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

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

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

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

4.6.2 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. Timeliness parameters are included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

4.7 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
3	(o)	Attribute:	User Description
4	(o)	Attribute:	Object Revision

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.

User description

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

Object revision

This optional attribute specifies the revision level of the object. It is a structured attribute composed of major and minor revision numbers. If Object Revision is supported, it contains both a Major Revision and a Minor Revision with a value range 0 to 15 for each. The use of major/minor fields is intended to provide the following features:

Major revision

The Major Revision field contains the major revision value for the object. A change to the major revision indicates that interoperability is affected by the change.

Minor revision

The Minor Revision field contains the minor revision value for the object. A change to the minor revision indicates that interoperability was not affected by the change -- that is users of the object will continue to be capable of interoperating with the object when its minor revision is changed, provided that the major revision remains the same.

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

FAL ASE/FAL class

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.

Additional detail

This optional parameter contains user data that accompanies the negative response. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive.

4.9 APDU size

APDU size is Communication Model dependent.

5 Type 4 communication model specification

5.1 Concepts

5.1.1 Overview

The concepts of this model basically follow the common concepts with a few exceptions. The basic principle described in 4.2.2.1 through 4.2.2.3 are applicable for this clause. This Type supports Gateway functionality within the application layer. Part of this can also be recognised from the common concepts. However, to promote readability, some elements from the Clause 4 are repeated in this clause.

5.1.2 Application entities

5.1.2.1 Definition of FAL AE

An FAL AE provides a set of services to enable communications between APs in a fieldbus environment. These services enable a user application in a Client to access user objects in a server.

The functionality of the FAL AE can be divided into two groups.

Application Relation (AR). The AR functionality is responsible for arranging the transportation of APDUs via the network.

Variable Access (VA). The Variable Access functionality is responsible for coding and encoding of APDUs. In a server the VA also convey data directly to / from user objects. How this is done is a local matter and out of scope for this standard.

5.1.2.2 FAL services

The services offered by the FAL AE to the user application are called FAL Services. Services for the AR are called AR ASE services and services for the Variable Access are called Route Endpoint ASE services (REP ASE services).

The FAL services provide access to local AR objects, and provide conveyance of requests and responses for access to real objects modelled as FAL accessible objects. The FAL provides as well confirmed as unconfirmed services. Service requests are conveyed to the AP containing the real object.

Figure 13 illustrates the FAL AE and the architectural relationships.

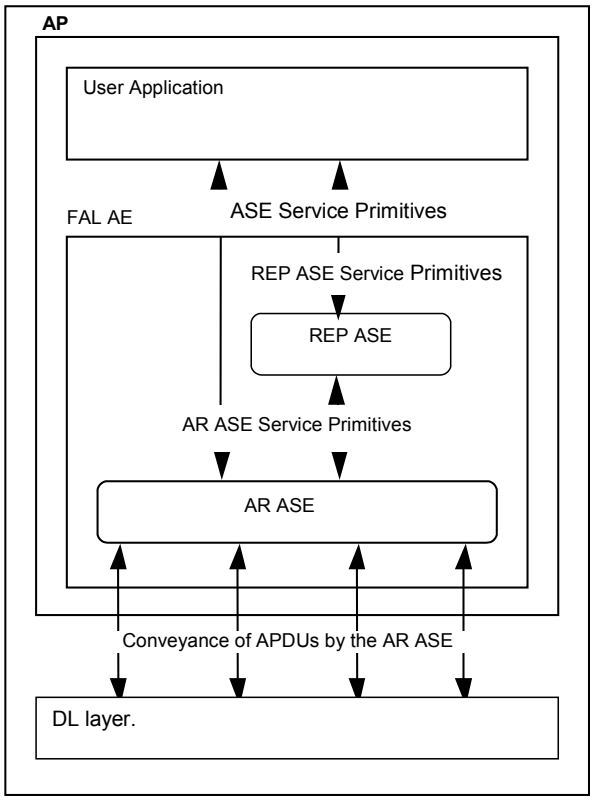

Figure 13 – FAL AE

5.1.2.3 AR ASE

To support access to the remote AP, the Application Relationship ASE is defined. It provides services to the AP for accessing communication-related parameters, and it provides services to the REP ASE for conveying service requests and responses.

The AR ASE provides services at the endpoints of ARs (AREPs).

5.1.2.4 REP ASE

The REP ASE provides a set of services used to read and write network visible attributes of Variable Objects and their application data. As only one ASE type is used remotely, the type is implicit and not transferred.

5.1.2.5 FAL service conveyance and gateway

The FAL AE provides services to convey requests and responses between User Applications and Variable Objects including Gateway functionality.

In a distributed system, application processes communicate with each other by exchanging application layer messages. The AR is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. Connectionless data link services are used for the exchanges. The data link layer transfers may be either acknowledged or unacknowledged.

5.1.2.6 Application relationships

5.1.2.6.1 Definition of AR

The AR is responsible for conveying messages between applications. The messages that are conveyed are FAL service requests and responses. Each service request and response is submitted to the AR ASE for transfer. Within the AR only one object type exist, the AR-Endpoint (AREP).

5.1.2.6.2 AR-endpoints

An AR-Endpoint (AREP) represents a service access point to a DLE within the AE. The AR Send Service conveys APDU and route information to the AREP and implicit to the DLE. The AREP is a system management object presenting the user configurable parameters of the related DLE.

This Type has no special system management services, because the attributes of the AREP objects are declared as Variable Objects by the user application. This makes the attributes network visible. The AR Get Attribute and AR Set Attribute services are used by these Variable Objects to access the attribute values.

When an AREP receives an APDU from its DLE (User-triggered) it conveys the APDU to the REP ASE or an AREP (gateway).

5.1.3 Gateway and routing

5.1.3.1 Overview

This Type supports gateway functionality. To identify the destination of a request, a route description is used. This route description holding a complete list of points to pass is sent with the request. On its way to the destination, the route evolves in such a way that it always contains the route back to the requester and the route to the destination. The elements in the route are endpoint and DL addresses.

5.1.3.2 Route endpoint

Within the REP ASE, Route Endpoint objects (REPs) represent endpoints, holding a Route description. An REP represents the starting point for conveying a message. An REP also represents the destination of the APDU.

5.1.3.3 Identification of endpoints

The endpoints are identified by endpoint addresses. These EP addresses are numerical Addresses, taken within an AE from the same name space of numbers, to achieve a unique identification. These EP addresses are used to identify AREPs and REPs.

5.1.3.4 DL addresses

The DLE is identified from the Link by a DL-address. DL-addresses are defined in IEC 61158-4-4, but as they are part of the route, they are mentioned here. There is a one to one relation between the DL-address of a DLE and the endpoint address of the corresponding AREP.

5.1.3.5 Route

The complete Route contains a Destination-Route and a Source-Route.

The Destination-Route describes how to reach the destination REP. It is a sequence of Endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the receiving DLE, AREP or REP.

The Source Route in the same way describes how to find the way back to the endpoint that initiated the request. It is a sequence of endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the transmitting DLE, AREP or REP.

5.1.3.6 Destination /source route conversion.

A Destination-Route conveyed by an REP is starting with the endpoint address of the local AREP. The next element is the DL-address of the receiving DLE. If the route goes through gateways a pair of addresses shall be added for each gateway, a DL-address and an AREP address. The final address is the endpoint address of the destination REP. The Source address is the endpoint address of the source REP.

The AR Send service removes the first element of the Destination-Route. The DLE inserts its own DL-address in front of the current Source Route before conveying the request to the link.

When a DLE receives an indication from the link, it removes the first element of the Destination Route (its own DL-address). The corresponding AREP inserts its own AREP address in front of the current Source Route before conveying the request to a REP or an AREP.

At the destination REP the Destination-Route holds the endpoint address of the REP, and the Source Route holds the complete route back to the requesting REP.

If any error appears with a request on its way to the destination, the current Source Route can be used for returning an error response.

5.1.4 Architecture summary

A summary of the FAL architecture is presented in Figure 14. It illustrates the major components of the FAL architecture and how they relate to each other.

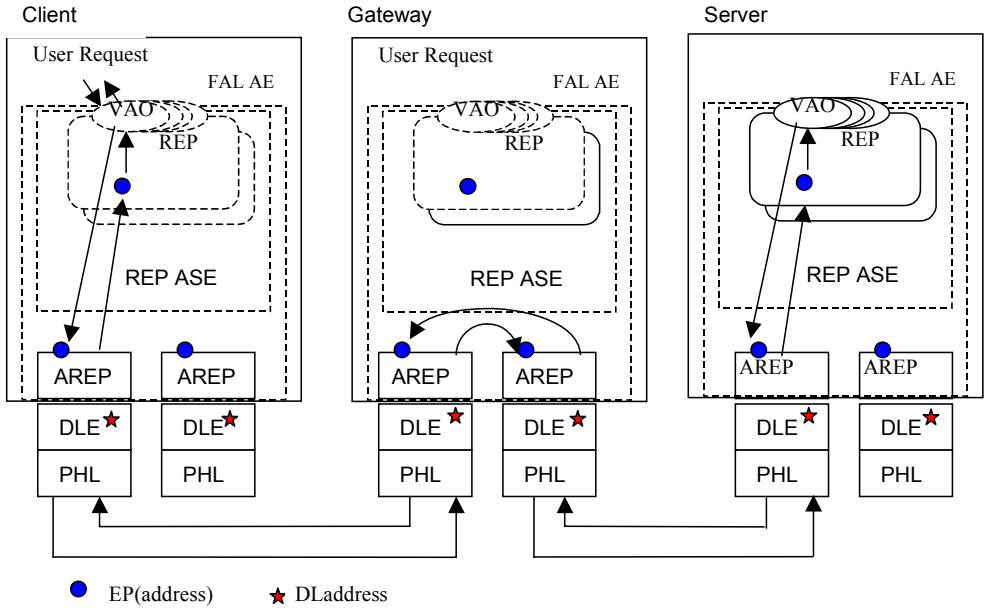

Figure 14 – Summary of the FAL architecture

5.1.5 FAL service procedures and time sequence diagrams

5.1.5.1 Overview

This subclause describes service procedures for FAL confirmed and unconfirmed services. An overview of the FAL service procedures is given in Figure 15.

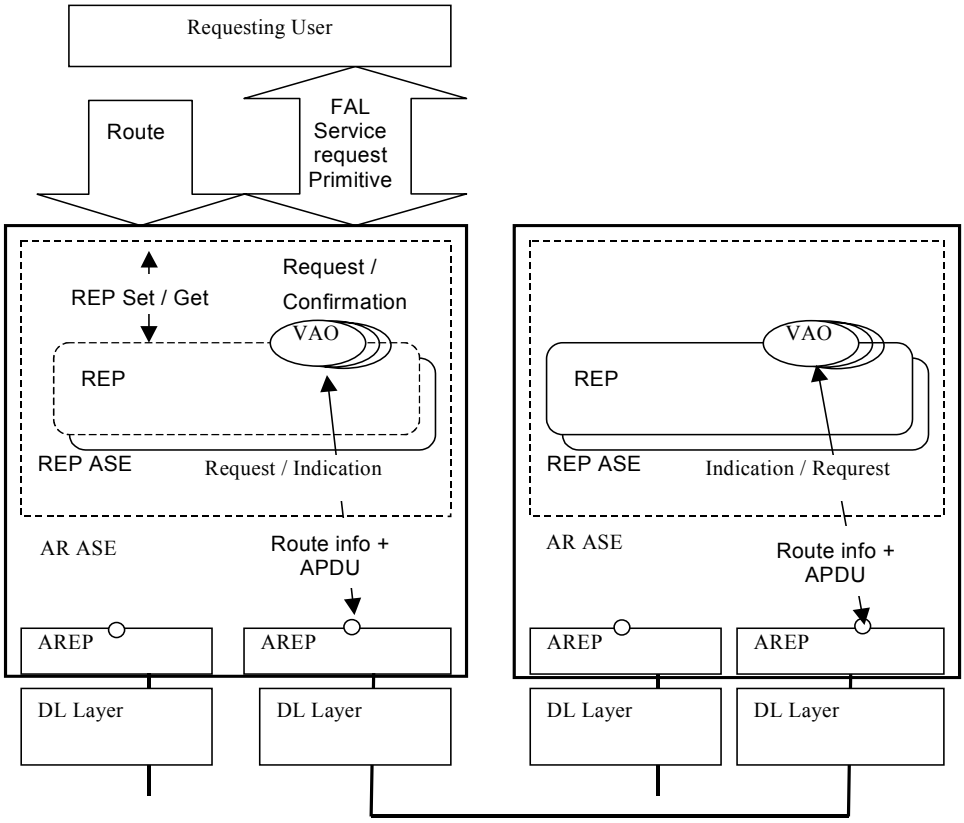

Figure 15 – FAL service procedure overview

5.1.5.2 FAL confirmed service procedures

At requesting AP:

The user application uses the REP ASE service "Reserve REP" to reserve a REP. The user application then uses the REP ASE service "Set REP Attribute" to set attributes of the REP, including the Destination Route to the remote REP.

The requesting user submits a confirmed request primitive to the REP ASE with the REP address as a parameter. The REP ASE builds the related confirmed request APDU and conveys it on the AREP, using an AR Send request primitive. The AREP address is indicated by the first element of the Destination route. The REP ASE starts an associated timer to monitor the request.

The AREP builds a request DLSDU and submits the route info together with the DLSDU to its DLE. The DLE sends the DLSDU at the first opportunity.

At responding server AP:

Upon receipt of the DLSDU, the receiving AREP builds an APDU and delivers the APDU and the received Route to the addressed REP by an AR Send indication primitive.

The REP ASE handles the indication by data exchange with the addressed Variable Object, builds the related unconfirmed response APDU and conveys it on the AREP, using an AR Send request primitive. The received Source route is used as Destination route for the response.

The AREP builds a request DLSDU and submits the route info, together with the DLSDU to its DLE. It is the task of the DL layer to decide whether to “queue” the DLSDU or send it immediately.

If the REP ASE is not able to process the indication and respond within “MaxIndicationDelay”, the REP ASE shall submit an AR Acknowledge request primitive to the receiving AREP, and in this way free the Token. The AR Acknowledge is submitted to the DLE.

At requesting AP receiving the response:

Upon receipt of the response DLSDU, the receiving AREP conveys the response to the REP ASE. The destination address conveyed to the REP ASE holds the endpoint address of the requesting REP. The REP object can now deliver the response to the requesting user application. This is accomplished by the user invocation of the REP ASE RESPONSE Service.

If the timer expires before the requesting REP has received the response, the REP cancels the associated transaction state machine and delivers the Error response “No answer” to the user application.

5.1.5.3 Confirmed service time sequence diagram

Figure 16 shows the time sequence diagram for the confirmed services.

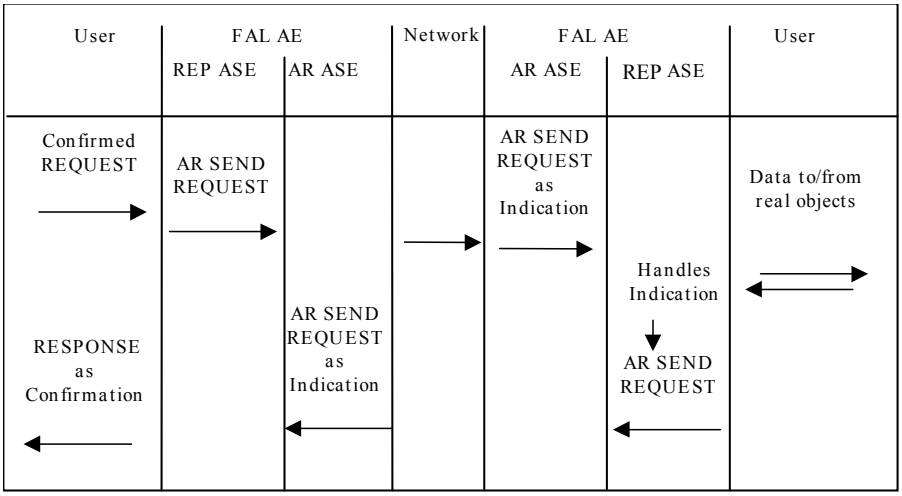

Figure 16 – Time sequence diagram for the confirmed services

5.1.5.4 Unconfirmed request service procedure

At requesting AP:

The user application uses the REP ASE service "Reserve REP" to reserve a REP. The user application then uses the REP ASE service "Set REP Attribute" to set attributes of the REP, including the Destination Route to the remote REP.

The requesting user submits an unconfirmed request primitive to the REP ASE with the REP address as a parameter. The REP ASE builds the related unconfirmed request APDU and conveys it on the AREP, using an AR Send request primitive. The AREP address is indicated by the first element of the Destination route.

The AREP builds a request DLSDU and submits the route info together with the DLSDU to its DLE. The DLE sends the DLSDU at the first opportunity.

At responding server AP:

Upon receipt of the DLSDU, the receiving AREP builds an APDU and delivers that and the received Route to the addressed REP by an AR Send indication primitive.

The REP ASE handles the indication by data exchange with the addressed Variable Object.

5.1.5.5 Unconfirmed service time sequence diagram

Figure 17 shows the time sequence diagram for the unconfirmed services.

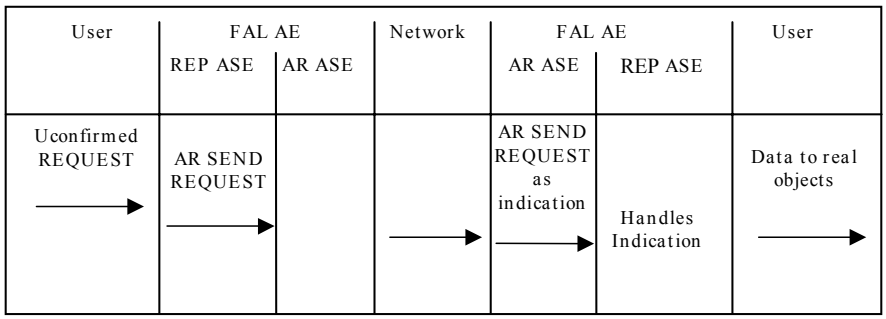

Figure 17 – Time sequence diagram for unconfirmed services

5.1.5.6 Confirmed / unconfirmed service procedure through gateways

Upon receipt of a DLSDU, the receiving AREP delivers an AR Send indication primitive to the local destination. If the endpoint address is a REP address, the APDU + Route is delivered to the REP ASE. If the endpoint address is an AREP address, the AR send service is used to pass it on to the network (Gateway).

For confirmed service Request, an AR Acknowledge is submitted to the “source AREP” unless the response is expected within “MaxIndicationDelay”. “MaxIndicationDelay" is an attribute of the "source AREP".

5.2 Variable ASE

5.2.1 Variable types

5.2.1.1 Overview

The Variable ASE specifies the machine independent syntax for application data conveyed by FAL services. The application layer supports the transfer of both basic and constructed variables. The encoding rules for basic variables and constructed variables as specified in this subclause are provided in IEC 61158-6-4.

Variable Objects (VAOs) are defined as instances of a variable class. The Variable ASE provides access to VAOs. VAOs are identified by a numeric Variable Identifier, and are instances of a variable class.

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 free.

Constructed types are packed according to the encoding rules for constructed types. The structure of a constructed type is not network visible. The structure of a constructed type is generated by a compiler and transferred by other means.

All variable classes have the common attributes Variable Identifier, Data Type Identifier, Octet length, Read enable, Write enable and Write protected. All variable classes have the common services Read, Write, Get Variable Attribute and Set Variable Attribute.

This standard recommends using floating point representation of measurement values scaled in SI units.

5.2.1.2 Basic variable types

Basic variable types have a constant length. The length of basic variable types is an integral number of octets. Defining new basic variable types is accomplished by defining new basic variable type classes.

5.2.1.3 Constructed variable types

5.2.1.3.1 Overview

Constructed variable types are needed to completely convey the variety of information present on the fieldbus. Four kinds of constructed variable types are defined in four classes in this standard: Complex, String, BitString and FIFO. Defining new kinds of constructed variable types is accomplished by defining new constructed variable type classes.

On the fieldbus a complete constructed variable can be transferred, or a part of it, e.g. a single field of a complex variable or an element of a string. When transferring only part of a variable, addressing with offset and maybe Bit-no is used. Offset indicates the distance (in octets) from the first octet of the variable to the part to be transferred, Bit-no indicates the bit number in one octet.

5.2.1.3.2 Complex variable type

A Complex variable type defines Arrays and Structures.

An Array is composed of an ordered set of homogeneously typed elements. This standard places no restriction on the 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.

A Structure is made of an ordered set of heterogeneously typed elements called fields. Like Arrays, this standard does not restrict the type of fields. However, the fields within a Structure can be of different types.

5.2.1.3.3 String variable type

A String is composed of an ordered set of an Actual Number of Elements field and a number of homogeneously typed elements. Once defined, the maximum number of elements in a String may not be changed.

5.2.1.3.4 BitString variable type

A BitString is defined as a series of bits. Once defined, the number of elements in a BitString may not be changed. The octet length of a BitString is the integral part of ((the number of elements + 7) divided by 8).

5.2.1.3.5 FIFO variable type

A FIFO queue is composed of a set of homogeneously typed elements. This standard places no restriction on the type of FIFO elements, but it does require that each element be of the same type. The first written element will be the first element that can be read. On the fieldbus only one, complete element can be transferred as a result of one service invocation.

5.2.2 Variable model class specification

5.2.2.1 Variable model

5.2.2.1.1 Formal model

The Variable formal model defines the characteristics common to all Variable classes. This class is not capable of being instantiated. It is present only for the inheritance of its attributes and services of its subclasses.

FAL ASE:			**VARIABLE ASE**
CLASS:	VARIABLE		
CLASS ID:			1
PARENT CLASS:			TOP
ATTRIBUTES:			
1	(m)	Key Attribute:	Variable Identifier
2	(o)	Attribute:	VariableTypeIdentifier
3	(o)	Attribute:	Octet Length
4	(o)	Attribute:	Read enable
5	(o)	Attribute:	Write enable
6	(o)	Attribute:	Write protected
SERVICES:			
1	(m)	OpsService:	Read
2	(m)	OpsService:	Write
3	(o)	OpsService:	Get Variable Attribute
4	(o)	OpsService:	Set Variable Attribute

5.2.2.1.2 Attributes

Variable identifier

This attribute identifies the Variable Object instantiating the variable type class.

VariableTypeIdentifier

This optional attribute identifies the variable type as a Boolean, Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Float32, Float64, UNICODE Char, Array, String, Structure or FIFO. The VariableTypeIdentifier is a numerical identifier, defined in IEC 61158-6. In this standard, a descriptive name corresponding to the identifier is stated.

Octet length

This optional attribute indicates the length in octets of the data of the Variable Object. For all types except FIFO octet length indicates the total number of octets. For FIFO types, octet length indicates the number of octets in one element.

Read enable

The value of this optional attribute indicates, whether the data value of the Variable Object can be read via the fieldbus.

Write enable

The value of this optional attribute indicates, whether the data value of the Variable Object can be updated via the fieldbus.

Write protected

The value of this optional attribute indicates, whether the data value of the Variable Object can only be updated via the fieldbus under certain conditions. The conditions for permitting update of a Write protected VAO are out of scope of this standard.

5.2.2.1.3 Services

Read

This service is used to read the data value of the Variable Object.

Write

This service is used to update the data value of the Variable Object.

Get variable attribute

This service is used to read an attribute of the Variable Object.

Set variable attribute

This service is used to update an attribute of the Variable Object.

5.2.3 Basic variable type specifications

5.2.3.1 Boolean variable type model

5.2.3.1.1 Formal model

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

FAL ASE:			**VARIABLE ASE**
CLASS:	BOOLEAN VARIABLE TYPE		
CLASS ID:			2
PARENT CLASS:			VARIABLE
ATTRIBUTES:			
SERVICES:			
1	(m)	OpsService:	And
2	(m)	OpsService:	Or
3	(m)	OpsService:	Test-And-Set

5.2.3.1.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Boolean
3	Octet Length	= 1

5.2.3.1.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation.

5.2.3.2 Integer8 variable type model

5.2.3.2.1 Formal model

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

FAL ASE: **VARIABLE ASE**
CLASS: INTEGER8 VARIABLE TYPE
CLASS ID: 3
PARENT CLASS: VARIABLE
ATTRIBUTES:
SERVICES:
1 (m) OpsService: And
2 (m) OpsService: Or
3 (m) OpsService: Test-And-Set

5.2.3.2.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:
2 VariableTypeIdentifier = Integer8
3 Octet Length = 1

5.2.3.2.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation.

5.2.3.3 Integer16 variable type model

5.2.3.3.1 Formal model

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

FAL ASE: **VARIABLE ASE**
CLASS: INTEGER16 VARIABLE TYPE
CLASS ID: 4
PARENT CLASS: VARIABLE
ATTRIBUTES:
SERVICES:
1 (m) OpsService: And
2 (m) OpsService: Or

5.2.3.3.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:
2 VariableTypeIdentifier = Integer16

3 Octet Length = 2

5.2.3.3.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

5.2.3.4 Integer32 variable type model

5.2.3.4.1 Formal model

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

FAL ASE: **VARIABLE ASE**
CLASS: INTEGER32 VARIABLE TYPE
CLASS ID: 5
PARENT CLASS: VARIABLE
ATTRIBUTES:
SERVICES:
1 (m) OpsService: And
2 (m) OpsService: Or

5.2.3.4.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2 VariableTypeIdentifier = Integer32
3 Octet Length = 4

5.2.3.4.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

5.2.3.5 Unsigned8 variable type model

5.2.3.5.1 Formal model

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

FAL ASE: **VARIABLE ASE**
CLASS: UNSIGNED8 VARIABLE TYPE
CLASS ID: 6
PARENT CLASS: VARIABLE
ATTRIBUTES:
SERVICES:

1	(m)	OpsService:	And
2	(m)	OpsService:	Or
3	(m)	OpsService:	Test-And-Set

5.2.3.5.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Unsigned8
3	Octet Length	= 1

5.2.3.5.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation.

5.2.3.6 Unsigned16 variable type model

5.2.3.6.1 Formal model

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

FAL ASE:			**VARIABLE ASE**
CLASS:	UNSIGNED16 VAIABLE TYPE		
CLASS ID:			7
PARENT CLASS:			VARIABLE
ATTRIBUTES:			
SERVICES:			
1	(m)	OpsService:	And
2	(m)	OpsService:	Or

5.2.3.6.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Unsigned16
3	Octet Length	= 2

5.2.3.6.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

5.2.3.7 Float32 variable type model

5.2.3.7.1 Formal model

This type has a length of four octets. The format for Float32 is that defined by IEC 60559 as single precision.

FAL ASE: **VARIABLE ASE**
CLASS: FLOAT32 VARIABLE TYPE
CLASS ID: 8
PARENT CLASS: VARIABLE
ATTRIBUTES:
SERVICES:

5.2.3.7.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2 VariableTypeIdentifier = Float32
3 Octet Length = 4

5.2.3.8 Float64 variable type model

5.2.3.8.1 Formal model

This type has a length of eight octets. The format for Float64 is that defined by IEC 60559 as double precision.

FAL ASE: **VARIABLE ASE**
CLASS: FLOAT64 VARIABLE TYPE
CLASS ID: 9
PARENT CLASS: VARIABLE
ATTRIBUTES:
SERVICES:

5.2.3.8.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2 VariableTypeIdentifier = Float64
3 Octet Length = 8

5.2.3.9 UNICODE char variable type model

5.2.3.9.1 Formal model

This type has a length of two octets. It is defined as a single UNICODE character.

FAL ASE: **VARIABLE ASE**
CLASS: UNICODE CHAR VARIABLE TYPE
CLASS ID: 10
PARENT CLASS: VARIABLE
ATTRIBUTES:
SERVICES:

5.2.3.9.2 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= UNICODE Char
3	Octet Length	= 2

5.2.4 Constructed variable type specifications

5.2.4.1 Complex variable type model

5.2.4.1.1 Formal model

A Complex Variable is an Array or a Structure.

An Array is composed of an ordered set of homogeneously typed elements. This standard places no restriction on the 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.

A Structure is made of an ordered set of heterogeneously typed elements called fields. Like Arrays, this standard does not restrict the type of fields. However, the fields within a Structure can be of different types.

FAL ASE:			**VARIABLE ASE**
CLASS:	COMPLEX VARIABLE TYPE		
CLASS ID:			11
PARENT CLASS:			VARIABLE
ATTRIBUTES:			
SERVICES:			
1	(m)	OpsService:	And
2	(m)	OpsService:	Or
3	(m)	OpsService:	Test-And-Set

5.2.4.1.2 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2	VariableTypeIdentifier	= Complex

5.2.4.1.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation. The operation is legal on subelements of integer or boolean type only.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation. The operation is legal on subelements of integer or boolean type only.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation. The operation is legal on subelements of Integer8, Unsigned8 or boolean type only.

5.2.4.2 String variable type model

5.2.4.2.1 Formal model

A String is composed of an ordered set of an Actual Number of Elements field and a number of homogeneously typed elements. Once defined, the maximum number of elements in a String may not be changed.

FAL ASE:			**VARIABLE ASE**
CLASS:	STRING VARIABLE TYPE		
CLASS ID:			12
PARENT CLASS:			VARIABLE
ATTRIBUTES:			
SERVICES:			
1	(m)	OpsService:	And
2	(m)	OpsService:	Or
3	(m)	OpsService:	Test-And-Set

5.2.4.2.2 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2 VariableTypeIdentifier = String

5.2.4.2.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation. The operation is legal on subelements of integer or boolean type only.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation. The operation is legal on subelements of integer or boolean type only.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation. The operation is legal on subelements of Integer8, Unsigned8 or boolean type only.

5.2.4.3 BitString variable type model

5.2.4.3.1 Formal model

A BitString is defined as a series of bits. Once defined, the number of bits in a BitString may not be changed. The octet length of a BitString is the integral part of ((the number of bits + 7) divided by 8).

FAL ASE:			**VARIABLE ASE**
CLASS:	BITSTRING VARIABLE TYPE		
CLASS ID:			13
PARENT CLASS:			VARIABLE
ATTRIBUTES:			
SERVICES:			
1	(m)	OpsService:	And
2	(m)	OpsService:	Or
3	(m)	OpsService:	Test-And-Set

5.2.4.3.2 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2 VariableTypeIdentifier = BitString

5.2.4.3.3 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation. The operation is legal on a subelement (which is a bit) only.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation. The operation is legal on a subelement (which is a bit) only.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation. The operation is legal on a subelement (which is a bit) only.

5.2.4.4 FIFO variable type model

5.2.4.4.1 Formal model

A FIFO queue is composed of a set of homogeneously typed elements. This standard places no restriction on the type of FIFO elements, but it does require that each element be of the same type. The first written element will be the first element that can be read. On the fieldbus only one, complete element can be transferred as a result of one service invocation.

FAL ASE:			**VARIABLE ASE**
CLASS:	FIFO VARIABLE TYPE		
CLASS ID:			14
PARENT CLASS:			VARIABLE
ATTRIBUTES:			
1	(m)	Attribute:	Next Element In
2	(m)	Attribute:	Next Element Out
3	(m)	Attribute:	Free elements
4	(m)	Attribute:	Used elements
5	(o)	Attribute:	ReRead
6	(o)	Attribute:	ReWrite

5.2.4.4.2 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2 VariableTypeIdentifier = FIFO

Octet length

The octet length of a FIFO indicates the length of a single element.

Next element in

Next Element In is an Integer16, which holds the index number for the next element to be written to the FIFO.

Next element out

Next Element Out is an Integer16, which holds the index number for the next element to be read from the FIFO.

Free elements

This attribute is an Integer16 which holds the current number of free elements in the FIFO (= 0 if full).

Used elements

This attribute is an Integer16 which holds the current number of used elements in the FIFO (= 0 if empty).

Reread

Reread is a Boolean. When the value of Reread is TRUE, a Read will return the previously read element, instead of returning and removing the element indexed by Next Element Out. This facility can be used if a transmission error resulted in a faulty read of an element.

Rewrite

Rewrite is a Boolean. When the value of Rewrite is TRUE, a Write will overwrite the previously written element, instead of inserting a new element. This facility can be used if a transmission error resulted in a faulty write of an element.

5.2.4.4.3 Services

Overview

The services defined in the Variable ASE provide access to attributes and data of Variable Objects. The services are invoked by a REP ASE REQUEST Service invocation. The REP ASE REQUEST Service conveys the Variable ASE Service parameters. Refer to 5.3.3 for the documentation of these parameters.

Read

This service is used to read (part of) the specified variable. No data is passed with the request. The value of the specified variable is returned in the response.

Write

This service is used to update (part of) the specified variable. The data passed with the request is written into (part of) the specified variable. No data is returned in the response.

And

This service performs a logical AND of (part of) the specified variable and the data passed with the request. The result is stored in (part of) the variable. The service is used to clear individual bits in a variable without first reading the variable and afterwards writing the result back to the variable. No data is returned in the response.

Or

This service performs a logical OR of (part of) the specified variable and the data passed with the request. The result is stored in (part of) the variable. The service is used to set individual bits in a variable without first reading the variable and afterwards writing the result back to the variable. No data is returned in the response.

Test-And-Set

This service performs a logical AND of (part of) the specified variable and the data passed in the request. The result of the AND operation is returned in the response. Then a logical OR operation of (part of) the specified variable and the data passed in the request is performed. The result of the OR operation is stored in (part of) the variable. The service is used to set individual bits in a variable, and in the same transmission read their state prior to the operation.

Get variable attribute

This service is used to read an attribute of the specified Variable Object. The attribute is specified by the offset parameter.

Set variable attribute

This service is used to update an attribute of the specified Variable Object. The attribute is specified by the offset parameter.

5.2.5 Route endpoint ASE

5.2.5.1 Overview

In the fieldbus environment, application processes contain data that remote applications are able to read and write. The data is accessed by means of the REP ASE. This ASE offers services to user applications in clients. The services REQUEST and RESPONSE give the access to real Variable Objects (VAOs) located in a server. In the server the REP ASE conveys data to / from the real Variable Object and returns responses. The services Get REP Attribute, Set REP Attribute, Reserve REP and Free REP are all for local use.

Route endpoint
Description

The Route Endpoint class is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. It uses AR ASE services for conveying APDUs.

An REP is an object container, containing Variable Objects of a variable type. It performs the access to variable objects. The Variable Object ID is unique within an REP. The request service addresses a Variable Object within the REP. The Variable Object ID is one of the request service parameters.

An REP in the proxy container role represents a remote REP in a Server. The remote REP is in the real container role. The Destination Route attribute of the REP in the client holds the Route to the real container REP, which it represents.

The REP ASE handles the segmentation when the data length exceeds the MaxDataSize.

When data is segmented, the order of segments is indicated by the offset part of the APDU.

To initiate access of a proxy Variable Object, the user application uses Reserve REP to get a REP currently not in use.

The attributes of the REP are set, including the Destination Route to the remote REP. This is done by the Set REP Attribute service. The requesting user then issues a REQUEST primitive with the REP ID and a Variable Object ID as parameters. The REP ASE builds a request APDU, and passes that along with Route Info in an AR Send request.

Following a successful transmission, if the REQUEST was confirmed, the AREP receives a response APDU and conveys that to the requesting REP by an AR Send indication. The user application can now read the result by issuing a RESPONSE primitive, with the REP ID as a parameter.

5.2.5.2 Route endpoint model

Formal Model

FAL ASE: **Route Endpoint ASE**

CLASS: Route Endpoint

CLASS ID: 1

PARENT CLASS: TOP

ATTRIBUTES:

1.	(m)	Key Attribute:	Endpoint Address
2.	(m)	Attribute:	Role (Proxy object / Real object)
3.	(m)	Attribute:	REP State
4.	(m)	Attribute:	Priority
5.	(m)	Attribute:	Confirmation
6.	(m)	Attribute:	Destination Route
7.	(m)	Attribute:	Source route
8.	(m)	Attribute:	Progress
9.	(m)	Attribute:	Capabilities
10.	(m)	Attribute:	Flat addressing

SERVICES:

1.	(m)	OpsService:	REQUEST
2.	(m)	OpsService:	RESPONSE
3.	(m)	OpsService:	Reserve REP
4.	(m)	OpsService:	Free REP
5.	(m)	OpsService:	Get REP Attribute
6.	(m)	OpsService:	Set REP Attribute

5.2.5.2.1 Attributes

5.2.5.2.1.1 Endpoint address

This key attribute holds the Endpoint address identifying the REP.

5.2.5.2.1.2 ROLE

This attribute specifies the role of the REP. The valid values are as follows.

Proxy container	Endpoints of this type are used for sending requests to servers and receive responses from them. The objects contained in REPs of this role are proxy objects for the real objects in the server.
Real container	Endpoints of this type are used for receiving confirmed and unconfirmed requests from clients and sending responses to them. The objects contained in REPs of this role are the real Variable Objects.

5.2.5.2.1.3 REP State

This attribute indicates the state of the REP. The values for this attribute are: IDLE RESERVED, WAITING FOR RESPONSE, RESPONSE RECEIVED or NOT IN USE.

5.2.5.2.1.4 Priority

The DLL may provide the possibility to send high priority APDUs before APDUs of lower priority. The priority only refers to the request on the local link, not requests passing gateways and not the response.

5.2.5.2.1.5 Confirmation

This attribute indicates whether the request has to be to confirmed or unconfirmed. The response is always returned unconfirmed. If the Destination Route contains one or more broadcast addresses (126) this attribute shall be set to unconfirmed.

5.2.5.2.1.6 Destination route

The Destination Route describes the Route to the destination REP. It is a sequence of Endpoint addresses and DL-addresses. On its way to the destination, the first part always indicates the address of the next DLE, AREP or REP to receive the APDU. In a request, the Destination Route holds the Route to the REP to respond. In a response, the Destination Route holds the Route to the requesting REP.

5.2.5.2.1.7 Source route

The Source Route describes the Route to the source REP. It is a sequence of endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the DLE, AREP or REP endpoint from where the APDU was conveyed. In a request, the Source Route holds the Route to the requesting REP. In a response, the Source Route holds the Route to the responding REP.

5.2.5.2.1.8 Progress

This attribute is only relevant in proxy container REPs. It indicates the progress of a request. It indicates number of segments that has been delivered divided with the total number of segments. This means for non-segmented requests the value is zero while waiting for the response, and one when the response is received. For segmented requests, the value will start from zero and gradually increase till it reaches the value of one.

5.2.5.2.1.9 Capabilities

This attribute defines the capabilities of the REP addressed by the Destination Route. It is a local attribute, which shall be set up by the user application, to reflect the capabilities of the REP containing the real Variable Objects. The value of Capabilities is used by the requesting REP to build the APDU. The attribute indicates, whether the responding REP is capable of handling bit addressing or not.

5.2.5.2.1.10 Flat addressing

This attribute indicates, whether the REP should be seen as a container of individual Variable Objects, or as a container of one, flat memory area. If Flat addressing is selected, the Variable Object ID parameter of the REQUEST service indicates the offset in octets from the beginning of the memory area.

5.2.6 Route endpoint ASE service specification

5.2.6.1 REQUEST service

5.2.6.1.1 Overview

This service is used to initiate access of the value of a Variable Object. It may be used for as well reading as updating the value. The result is the operation (if any) is retrieved by the RESPONSE service.

5.2.6.1.2 Service primitives

The service parameters for this service are shown in Table 1. The Indication parameters are indicating the parameters in the APDU received by the remote REP. These parameters are for local use within the remote REP and its variable objects.

Table 1 – REQUEST service parameters

Parameter name	Req	Ind	Cnf
Argument	M	M	
REP	M	M	
Variable object ID	M	M(=)	
Variable Service	M	M(=)	
Data length	M	M(=)	
Offset/Attribute	C	C(=)	
Bit-no	C	C(=)	
Data	C	C(=)	
Result			
Status			M

Argument

The argument carries the parameters of the REQUEST service invocation.

REP

This parameter is the EP address of the REP holding the route information.

Variable ID

This parameter identifies the Variable Object. The Variable ID indicates the Variable Object on which the service is to be performed.

Variable service

This parameter holds the Variable service to be performed. (Read, Write, And, Or, Test-And-Set, Get attribute and Set Attribute).

Data length

This parameter indicates the number of octets of data to be transferred.

Offset/attribute

This parameter is used when accessing only part of a structured variable or an Attribute of the variable. When accessing part of a structured variable, it indicates the offset in octets to the starting octet of the data block to be transferred, relative to the first octet of the variable. When accessing attributes, it identifies the attribute to be accessed.

Bit-no

This parameter is used to select a bit within a BitString. It indicates the bit number (1-8) within an octet. The Offset/Attribute parameter is used to select the octet.

Data

This parameter holds the data to be transferred.

Status

As a result of the request, Status is returned indicating OK or FAILURE.

5.2.6.2 RESPONSE service

5.2.6.2.1 Overview

This service is used to return the result of a confirmed REQUEST to the requesting user application in a requester.

5.2.6.2.2 Service primitives

The service parameters for this service are shown in Table 2. The response parameters are for local use within the responding REP and its variable object.

Table 2 – RESPONSE service parameters

Parameter name	Rsp	Cnf
Argument	M	M
REP	M	M
Result		
Variable Service	M	M
Data length	M	M(=)
Error status	M	M(=)
Data	C	C(=)

Argument

The argument carries the parameters of the service request.

REP

This parameter holds the ID of the REP holding the route information.

Variable service

This parameter holds the Variable service that has been performed. (Read, Write, And, Or, Test and Set, Get attribute and Set Attribute).

Data length

This parameter indicates the number of octets of data that have been transferred.

Error status

This parameter holds error information for errors occurred in any layer local or remote. For details, see Table 3.

The error code can be generated anywhere on the route between client and server, or by the responding Variable Object.

Table 3 – Error codes by source

Error description	Client	Gateway	Server
User Application:			
Instruction Error	X		X
Info length error	X		X
FIFO full or empty			X
Write protection			X
Data format error			X
Variable Object ID error			X
Time Out	X		
Actual data error			X
Historical data error			X
FAL:			
Route error	X	X	X
DL-layer:			
No Response	X	X	
WAIT TOO LONG	X	X	
Out of sync	X	X	
CRC ERROR	X	X	
DLE not Client	X	X	
Physical Layer:			
NET SHORTCIRCUIT	X	X	
OVERRUN/FRAMING ERROR	X	X	
RS-232 HANDSHAKE ERROR	X	X	X

Data

This parameter holds the data that has been transferred.

5.2.6.3 Reserve REP service

5.2.6.3.1 Overview

This service is used by the user application to reserve an REP currently not in use, and to get the ID of the reserved REP.

5.2.6.3.2 Service primitives

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

Table 4 – Reserve REP service parameters

Parameter name	Req	Cnf
Result		
REP		M

REP

This parameter holds the ID of the reserved REP. If no REP is available, zero is returned.

5.2.6.4 Free REP Service

5.2.6.4.1 Overview

This service is used by the user application to free an REP by changing the state of the REP to “not in use”.

5.2.6.4.2 Service primitives

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

Table 5 – Free AREP service parameters

Parameter name	Req
Argument	M
REP	M

Argument

The argument carries the parameters of the service request.

REP

This parameter holds the EP address of the reserved REP to free.

5.2.6.5 Get REP attribute service

5.2.6.5.1 Overview

This service is used by the user application to read an attribute of a REP locally.

5.2.6.5.2 Service Primitives

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

Table 6 – Get REP attribute service parameters

Parameter name	Req	Cnf
Argument	M	
REP	M	
Attribute index	M	
Result (+)		
Value		C
Result (-)		
Status		C

Argument

The argument carries the parameters of the service request.

REP

This parameter specifies the REP.

Attribute index

This parameter identifies the attribute to read.

Value

This parameter returns the value of the specified attribute.

Status

This parameter returns error information if the service fails. The possible values are: Illegal attribute index, Illegal REP address.

5.2.6.6 Set REP attribute service

5.2.6.6.1 Overview

This service is used by the user application to write a value into an attribute of a REP locally.

5.2.6.6.2 Service primitives

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

Table 7 – Set REP attribute service parameters

Parameter name	Req	Cnf
Argument	M	
REP	M	
Attribute index	M	
Result		
Status		M

Argument

The argument carries the parameters of the service request.

REP

This parameter specifies the REP.

Attribute index

This parameter identifies the attribute to update within the specified REP.

Value

This parameter holds the value to be written to the specified attribute.

Status

As a result of the request, Status is returned indicating OK or the reason for failure. The possible values are: OK, Illegal attribute index, Illegal value, Illegal REP address.

5.3 Application relationship ASE

5.3.1 Overview

To support access to the remote AP, the Application Relationship ASE is defined. It provides services to the AP for accessing communication-related parameters, and it provides services to the REP ASE for conveying service requests and responses.

The AR ASE provides services at the endpoints of ARs (AREPs).

5.3.2 Application relationship class specification

5.3.2.1 Application relationship formal model

The functionality of the AR ASE is described in 5.1.

The application ASE defines one class, the AREP class.

FAL ASE: **Application Relationship ASE**
CLASS: **AREP**
CLASS ID: 1
PARENT CLASS: TOP
ATTRIBUTES:

1.	(m)	Key Attribute:	Endpoint Address
2.	(m)	Attribute:	Role (Client, Server, Peer)
3.	(m)	Attribute:	DLL Reference
4.	(m)	Attribute:	MaxPDUSize
5.	(m)	Attribute:	MaxDataSize
6.	(m)	Attribute:	Acknowledgement
7.	(m)	Attribute:	MaxIndicationDelay
8.	(m)	Attribute:	Local DLE address
9.	(o)	Attribute:	MaxRetryTime
10.	(o)	Attribute:	MaxRetries
11.	(o)	Attribute:	MaxOutstandingRequests
12.	(o)	Attribute:	BaudRate
13.	(o)	Attribute:	NumberOfClientDLEs

5.3.2.1.1.1 SERVICES:

1	(m)	OpsService:	AR-Get Attribute
2	(m)	OpsService:	AR-Set Attribute
3	(m)	OpsService:	AR-Send
4	(m)	OpsService:	AR-Acknowledge

5.3.2.1.1.2 Endpoint address

This attribute specifies the numeric identifier of the AREP. It is used by the FAL to select the AREP, and implicitly the DLE for a request.

5.3.2.1.1.3 Role

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

Client Endpoints of this type send confirmed and unconfirmed Request APDUs to servers and receive Response APDUs.

Server Endpoints of this type receive confirmed and unconfirmed Request APDUs from clients and send unconfirmed Response APDUs.

Peer Endpoints of this type act as both Clients and Servers.

5.3.2.1.1.4 MaxPDUSize

This attribute specifies the maximum PDU size that can be sent by the DLL.

5.3.2.1.1.5 MaxDataSize

This attribute specifies the maximum data size that can be sent by the DLL. If the data length exceeds MaxDataSize, the VARIABLE ASE shall segment the request.

5.3.2.1.1.6 DLL reference

This attribute contains the necessary context to convey the DLL relations.

5.3.2.1.1.7 Acknowledgement

This attribute describes the type of Acknowledgement to be used by DLL: Acknowledged or Unacknowledged transfer of Confirmed APDUs, and Acknowledged or Unacknowledged transfer of Unconfirmed APDUs.

5.3.2.1.1.8 MaxIndicationDelay

This attribute indicates to the user application, how long time a Variable Object can use to prepare a response after receiving an indication requiring that. If the Variable Object is unable to prepare a response within MaxIndicationDelay, it shall issue an AR-Acknowledge. The value of MaxIndicationDelay is calculated by the DLE.

5.3.2.1.1.9 Local DLE address

This attribute reflects the DL address of the related DLE. In some situations it may be a read-only attribute.

5.3.2.1.1.10 MaxRetryTime

This attribute indicates the maximum time the DLE should try to re-transmit a request as a result of Acknowledge responses from the remote Variable Object.

5.3.2.1.1.11 MaxRetries

This attribute indicates the maximum number of re-transmissions carried out by the DLE as a result of transmission errors.

5.3.2.1.1.12 MaxOutstandingRequests

This attribute indicates the maximum number of requests that the AREP may initiate without receiving the related responses.

5.3.2.1.1.13 BaudRate

This attribute specifies the baud rate used by the physical layer. It is conveyed to/from the physical layers by the DLEs.

5.3.2.1.1.14 NumberOfClientDLEs

This attribute only relates to a DLE using the P-NET protocol in the client role. It specifies the number of participants in a token round.

5.3.3 Application relationship ASE service specifications

5.3.3.1 Overview

This subclause contains the definition of the services that are unique to this ASE. The services are the following:

AR Send

AR Acknowledge

AR Get Attribute

AR Set Attribute

5.3.3.2 Common parameter definition

Parameters used in more AR ASE services are defined below:

5.3.3.2.1.1 Route info

The complete DL Route contains the following elements:

5.3.3.2.1.2 Destination route

The Destination-Route describes how to reach the destination REP. It is a sequence of Endpoint addresses and DL-addresses. On its way to the destination, the first part always indicates the address of the next DLE, AREP or REP to receive the APDU.

5.3.3.2.1.3 Source route

The Source Route in the same way describes how to find the way back to the endpoint that initiated the request. It is a sequence of endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the DLE, AREP or REP endpoint from where the APDU was sent.

5.3.3.2.1.4 Priority

The underlying layer may provide the possibility to send high-priority APDUs before APDUs of lower priority. This priority only refers to the request on the local link, not requests passing gateways and not the response.

5.3.3.2.1.5 Confirmation

This parameter indicates whether the request has to be to confirmed or unconfirmed. A response is always returned unconfirmed. If the destination route contains one or more broadcast addresses (126) this attribute shall be set to unconfirmed.

5.3.3.2.1.6 APDU header

The APDU Header parameter holds a Control/Status subfield indicating the Variable Service, an APDU format subfield, indicating whether the APDU holds an offset, and finally an APDU length subfield, indicating the octet length of the APDU.

5.3.3.2.1.7 APDU body

This parameter holds the APDU Body to send or receive.

5.3.3.3 AR send service

5.3.3.3.1 Overview

The AR Send service is used to send confirmed or unconfirmed APDUs from one AREP to another.

It is the task of the receiving AREP to convey the APDU to the Endpoint specified by the first element in the destination route. In the gateway situation the AR is using this AR Send service to convey the APDU to an AREP.

5.3.3.3.2 Service primitives

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

Table 8 – AR send service parameters

Parameter name	Req	Ind	Cnf
Argument	M	M	
Route info	M	M	
APDU Header	M	M(=)	
APDU Body	C	C(=)	
Result			
Status			M

Argument

The argument carries the parameters of the service request.

Status

As a result of the request, Status is returned indicating OK or the locally detected reason for failure. Status only refers to local conditions. The possible values are: OK, Route Error.

5.3.3.4 AR acknowledge service

5.3.3.4.1 Overview

When the Variable ASE in a confirmed request is not able to process the indication and respond within the time limit, “MaxIndicationDelay”, it shall submit an AR Send Acknowledge request primitive to the local AREP, and in this way free the Token. “MaxIndicationDelay” is an attribute of the AREP.

5.3.3.4.2 Service Primitives

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

Table 9 – AR acknowledge service parameters

Parameter name	Req
Argument	M
Route info	M

Argument

The argument carries the parameters of the service request.

AR get attribute service

5.3.3.4.3 Overview

This confirmed service is used to read the value of attributes of an AREP locally.

5.3.3.4.4 Service parameters

The service parameters for the AR Get Attribute Service are shown in Table 10.

Table 10 – AR get attributes service parameters

Parameter name	Req	Cnf
Argument	M	
AREP	M	
Attribute index	M	
Result (+)		S
Value		C
Result (-)		S
Status		C

Argument

The argument carries the parameters of the service request.

AREP

This parameter identifies the AREP by its endpoint address.

Attribute index

This parameter identifies the attribute.

Result (+)

This selection type parameter indicates that the request succeeded.

Value

This parameter is returned with the requested attribute value if the request succeeded.

Result (-)

This selection type parameter indicates that the request failed.

Status

If the request failed, Status is returned indicating the reason for failure. The possible values are: Illegal attribute index, Illegal AREP address.

5.3.3.5 AR set attributes service

This confirmed service is used to set the current value of attributes of an AREP locally.

5.3.3.5.1 Service parameters

The service parameters for the AR Set Attribute Service are shown in Table 11.

Table 11 – AR set attributes service parameters

Parameter name	Req	Cnf
Argument	M	
AREP	M	
Attribute index	M	
Value	M	
Result		
Status		M

Argument

The argument carries the parameters of the service request.

AREP

This parameter identifies the AREP by its endpoint address. The legal values are 1 to 16 inclusive.

Attribute index

This parameter identifies the attribute to be updated.

Value

This parameter holds the new attribute value.

Status

As a result of the request, Status is returned indicating OK or the reason for failure. The possible values are: OK, Illegal attribute index, Illegal value, Illegal AREP address.

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 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*

———————

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
P.O. Box 131
CH-1211 Geneva 20
Switzerland

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