BS EN 61158-5-20:2014

BSI Standards Publication

Industrial communication networks — Fieldbus specifications

Part 5-20: Application layer service definition — Type 20 elements

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 61158-5-20:2014. It is identical to IEC 61158-5-20:2014. It supersedes BS EN 61158-5-20:2012 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications: process measurement and control, including fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

ISBN 978 0 580 79462 9
ICS 25.040.40; 35.100.70; 35.110

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 October 2014.

Amendments issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 61158-5-20

October 2014

ICS 25.040.40; 35.100.70; 35.110

Supersedes EN 61158-5-20:2012

English Version

Industrial communication networks - Fieldbus specifications - Part 5-20: Application layer service definition - Type 20 elements (IEC 61158-5-20:2014)

Réseaux de communication industriels - Spécifications des bus de terrain - Partie 5-20: Définition des services de la couche application - Eléments de type 20 (CEI 61158-5-20:2014)

Industrielle Kommunikationsnetze - Feldbusse - Teil 5-20: Dienstfestlegungen des Application Layer (Anwendungsschicht) - Typ 20-Elemente (IEC 61158-5-20:2014)

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Foreword

The text of document 65C/763/FDIS, future edition 3 of IEC 61158-5-20, prepared by SC 65C “Industrial networks” of IEC/TC 65 “Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61158-5-20:2014.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2015-06-22
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2017-09-22

This document supersedes EN 61158-5-20:2012.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

Endorsement notice

The text of the International Standard IEC 61158-5-20:2014 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61158-6-20	NOTE	Harmonized as EN 61158-6-20.
IEC 61784-1	NOTE	Harmonized as EN 61784-1.
IEC 61784-2	NOTE	Harmonized as EN 61784-2.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

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

NOTE 1 When an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu

Publication	Year	Title	EN/HD	Year
IEC 61158-1	2014	Industrial communication networks - Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series	EN 61158-1	2014
IEC 62591	2010	Industrial communication networks - Wireless communication network and communication profiles - WirelessHART^{TM}	EN 62591	2010
ISO/IEC 7498-1	-	Information technology - Open Systems Interconnection - Basic reference model: The basic model	-	-
ISO/IEC 8824-1	-	Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation	-	-
ISO/IEC 8859-1	-	Information technology - 8-bit single-byte coded graphic character sets - Part-1: Latin alphabet No. 1	-	-
ISO/IEC 9545	-	Information technology - Open Systems Interconnection - Application layer structure	-	-
ISO/IEC 10731	-	Information technology - Open Systems Interconnection - Basic Reference Model - Conventions for the definition of OSI services	-	-
IEEE 754	-	IEEE Standard for Floating-Point Arithmetic	-	-

CONTENTS

INTRODUCTION

This document is one of a series produced to facilitate the interconnection of automation system components. It is related to other documents in the set as defined by the “three-layer” fieldbus reference model described in IEC 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 document 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 document is a conceptual architectural service, independent of administrative and implementation divisions.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-20: Application layer service definition –

1 Scope

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 International 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 20 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 International Standard defines in an abstract way the externally visible service provided by the Type 20 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 International Standard is to define the services provided to the FAL user at the boundary between the user and the Application Layer of the Fieldbus Reference Model.

This International Standard specifies the structure and services of the IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI Application Layer Structure (ISO/IEC 9545).

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 International Standard to provide access to the FAL to control certain aspects of its operation.

2 Normative references

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

NOTE All parts of the IEC 61158 series, as well as IEC 61784-1 and IEC 61784-2 are maintained simultaneously. Cross-references to these documents within the text therefore refer to the editions as dated in this list of normative references.

IEC 61158-1:2014, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 62591:2010, *Industrial communication networks – Wireless communication network and communication profiles – WirelessHART™*

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

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

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

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

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

ANSI/IEEE 754: *IEEE Standard for Floating-Point Arithmetic*

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 ISO/IEC 7498-1 terms

a) abstract syntax
b) application entity
c) application process
d) application protocol data unit
e) application service element

3.1.2 ISO/IEC 9545 terms

a) application-entity-invocation
b) application-service-element
c) application-service-element

3.1.3 ISO/IEC 8824-1 terms

a) object identifier
b) type
c) value
d) simple type
e) structured type
f) component type

g) tag
h) true
i) false
j) integer type
k) octet string type
m) null type

3.2 IEC 61158-1 terms

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

3.2.1
application
function or data structure for which data is consumed or produced

3.2.2
application object
object class that manages and provides the run time exchange of messages across the network and within the network device

Note 1 to entry: Multiple types of application object classes may be defined.

3.2.3
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.2.4
application process object
component of an application process that is identifiable and accessible through an FAL application relationship

Note 1 to entry: 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

3.2.5
application process object class
class of application process objects defined in terms of the set of their network-accessible attributes and services

3.2.6
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

Note 1 to entry: This relationship is activated either by the exchange of application-protocol-data-units or as a result of pre-configuration activities

3.2.7
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 1 to entry: Each application process involved in the application relationship maintains its own application relationship endpoint

3.2.8
attribute
description of an externally visible characteristic or feature of an object

Note 1 to entry: 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

3.2.9
behaviour
indication of how the object responds to particular events

Note 1 to entry: Its description includes the relationship between attribute values and services.

3.2.10
class
set of objects, all of which represent the same kind of system component

Note 1 to entry: A class is a generalisation of the 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

3.2.11
class attributes
attribute that is shared by all objects within the same class

3.2.12
class code
unique identifier assigned to each object class

3.2.13
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service

Note 1 to entry: A class specific object is unique to the object class which defines it.

3.2.14
client
a) an object which uses the services of another (server) object to perform a task
b) an initiator of a message to which a server reacts, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

3.2.15
conveyance path
unidirectional flow of APDUs across an application relationship

3.2.16
cyclic
term used to describe events which repeat in a regular and repetitive manner

3.2.17
endpoint
one of the communicating entities involved in a connection

3.2.18
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.2.19
error code
identification of a specific type of error within an error class

3.2.20
management information
network-accessible information that supports managing the operation of the fieldbus system, including the application layer

Note 1 to entry: Managing includes functions such as controlling, monitoring, and diagnosing.

3.2.21
server

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

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

3.2.22
service
operation or function than an object and/or object class performs upon request from another object and/or object class

Note 1 to entry: A set of common services is defined and provisions for the definition of object-specific services are provided. Object-specific services are those which are defined by a particular object class to perform a required function which is not performed by a common service.

3.3 Type 20 fieldbus application-layer specific definitions

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

3.3.1
analog channel
continuously varying electrical signal connecting a field device to the remainder of the data acquisition or control system

Note 1 to entry: Some field devices support multiple analog channels (input or output). Each analog channel transmits a single dynamic variable to or from the field device.

3.3.2
broadcast
process of sending a PDU to all devices that are connected to the network and are able to receive the transmission

3.3.3
broadcast address
address used by a master to send a command to all devices

3.3.4
burst mode
initiation of communication activity by a slave device at cyclic interval without request from a master

3.3.5
comm error
detectable error in receiving a PhPDU or DLPDU, also ‘Communication error code’ octet of APDU

3.3.6
device
any entity containing an implementation of Type 20 fieldbus

3.3.7
device ID
serial number for a device that is unique among all instances of one type of device

Note 1 to entry: The manufacturer is required to assigned unique value for every device that has the identical values for Manufacturer ID and Device Type.

3.3.8
device type
manufacturer's type of a device, e.g. its product name

Note 1 to entry: The value of this attribute is unique among all manufacturers and all type of devices. Its value specifies the set of commands and data objects supported by the device.

3.3.9
device variable
uniquely defined data item within a Field Device that is always associated with the process-related information

Note 1 to entry: A device variable's value varies in response to changes and variations in the process to which the device is connected.

3.3.10
dynamic variable
device variable that is assigned as the dynamic variable and possibly associated with an analog channel

Note 1 to entry: A device may contain up to four variables – primary, secondary, tertiary, and quaternary variables. These are collectively called the dynamic variables.

3.3.11
expanded device type
manufacturer's type of the device

Note 1 to entry: The value of this attribute is unique among all manufacturers and all type of devices. Its value specifies the set of commands and data objects supported by the device.

3.3.12
field device
physical entity that is connected to the process or to plant equipment and has at least one signalling element that communicates with other signalling element(s) via the network

Note 1 to entry: It directly connects to the sensor or actuator or performs process control function and it is directly connected to the physical layer specified in this standard. It may generate or receive an analog signal in addition to a digital signal.

3.3.13
long tag
32 character restricted ISO Latin-1 string used to identify a field device

3.3.14
loop current
value measured by a milli-ammeter in series with the field device

Note 1 to entry: The loop current is a near DC analog 4-20 mA signal used to communicate a single value between the control system and the field device. Voltage mode devices use "Volts DC" as their engineering units where "loop current" values are used.

3.3.15
manufacturer ID
string identifying the manufacturer that produced the device

Note 1 to entry: A manufacturer is required to use the value assigned to it and is not permitted to use the value assigned to another manufacturer.

3.3.16
master
device that initiates communication activity by sending request PDU to a another device and expecting a response frame from that device

3.3.17
network
single pair of cable, connectors, associated signaling elements by which a given set of signaling devices are interconnected and non-signaling elements that are attached to the same pair of cable

Note 1 to entry: An installation using multiple-pair wire and a common network power supply is considered as multiple networks.

3.3.18
polling address
identifier assigned to a device such that it is unique within the network to which the device is connected

3.3.19
slave
device that initiates communication activity only after it receives a request PDU from a master device and it is required to send a response to that request

3.3.20
tag
8 character ASCII string used to identify the field device

3.3.21
unique ID
identifier assigned to a device which is unique among all instances of the devices compliant to this standard

3.4 Abbreviations and symbols

AE	Application Entity
AL	Application Layer
AP	Application Process
APDU	Application Protocol Data Unit
APO	Application Process Object
AR	Application Relationship
AREP	Application Relationship End Point
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Cnf	Confirmation
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLL	Data Link Layer

DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	DL-service-data-unit
FAL	Fieldbus Application Layer
ID	Identifier
IEC	International Electrotechnical Commission
Ind	Indication
OSI	Open Systems Interconnect
Req	Request
Rsp	Response
VFD	Virtual Field Device

3.5 Conventions

3.5.1 Overview

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

The class specification defines the attributes of the class. The service specification defines the services that are provided by the ASE.

3.5.2 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. The CLASS ID is not used in this document.

(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 document.

(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.

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

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

3.5.3 Conventions for service definitions

3.5.3.1 General

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

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

3.5.3.2 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.

This document uses a tabular format to describe the component parameters of the service primitives. The parameters that apply to each group of service primitives are set out in tables throughout the remainder of this document. Each table consists of up to six columns: a column for the name of the service parameter, and a column each for those primitives and parameter-transfer directions used by the service. The possible six columns are

1) the parameter name;
2) the request primitive's input parameters;
3) the request primitive's output parameters;

   NOTE 1 This is a seldom-used capability. Unless otherwise specified, request primitive parameters are input parameters.

4) the indication primitive's output parameters;
5) the response primitive's input parameters; and
6) the confirm primitive's output parameters.

   NOTE 2 The request, indication, response and confirm primitives are also known as requestor.submit, acceptor.deliver, acceptor.submit, and requestor.deliver primitives, respectively (see ISO/IEC 10731).

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.

3.5.3.3 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.

NOTE The IEC 61158-5 subseries define sets of abstract services. They are neither protocol specifications nor implementation specifications nor concrete programming interface specifications. Therefore there are restrictions on the extent to which service procedures can be mandated in the parts of IEC 61158-5 subseries. Protocol aspects that can vary among different protocol specifications or different implementations that instantiate the same abstract services are unsuitable for inclusion in these service definitions, except at the level of abstraction that is necessarily common to all such expressions.

For example, the means by which service providers pair request and reply PDUs is appropriate for specification in an IEC 61158-6 subseries protocol specification document but not in an IEC 61158-5 subseries abstract service definition document. Similarly, local implementation methods by which a service provider or service user pairs request and confirm(ation) primitives, or indication and response primitives, is appropriate for an implementation specification or for a programming interface specification, but not for an abstract service document or for a protocol document, except at a level of abstraction that is necessarily common to all embodiments of the specifying document. In all cases, the abstract definition is not permitted to over-specify the more concrete instantiating realization.

Further information on the conceptual service procedures of an implementation of a protocol that realizes the services of one of the IEC 61158-5 subseries abstract service definitions can be found in IEC 61158-1, 9.6.

4 Concepts

The common concepts and templates used to describe the application layer service in this document are detailed in IEC 61158-1, Clause 9.

5 Data type ASE

5.1 Overview

5.1.1 General

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

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

Data types are defined as instances of the data type class, as shown in Figure 1.

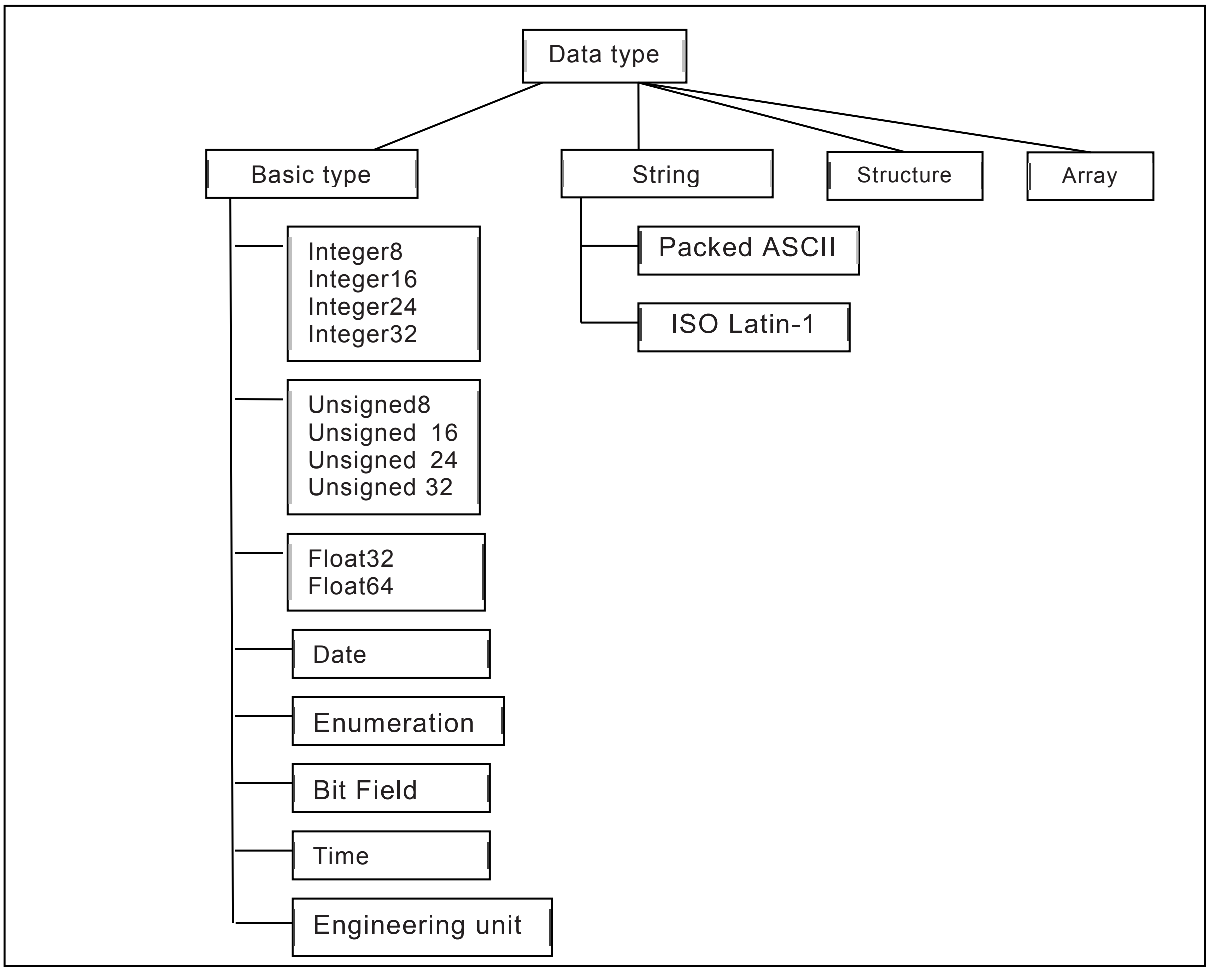

Figure 1 – Data type class hierarchy

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

The basic data classes are always fixed length data types. Standard types taken from ISO/IEC 8824-1 are referred to as *simple* data types. Other standard basic data types are defined specifically for Fieldbus applications and are referred to as *specific types*.

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

5.1.2 Basic types

Most basic types are defined from a set of ISO/IEC 8824-1 types (simple types). Some ISO/IEC 8824-1 types have been extended for Fieldbus specific use (specific types).

Simple types are ISO/IEC 8824-1 universal types. They are defined in this document to provide them with Fieldbus class identifiers.

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

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

NOTE Integer, Packed ASCII, ISO Latin-1 and Date are defined in this document for the purpose of assigning Fieldbus class identifiers to them. This document does not change their definitions as specified in ISO/IEC 8824-1.

5.1.3 Constructed types

5.1.3.1 Overview

Constructed data types are needed to completely convey the variety of information present on the Fieldbus. There are three kinds of constructed types defined for this document – string, array and structure.

5.1.3.2 String

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

5.1.3.3 Structure

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

5.1.3.4 Array

An array is composed of an ordered set of homogeneously typed elements. The data type of array elements can be fixed length basic type or structure. All elements of an array shall be of the same type.

5.1.3.5 Nesting level

This document permits structures and arrays to contain structures and arrays.

5.1.4 Specification of user defined data types

Users may find it necessary to define custom data types for their own applications. User defined types are not supported by this document.

5.1.5 Transfer of user data

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

The rules for encoding user data in FAL protocol data units is data type dependent. These rules are defined in IEC 61158-6-20. User-defined data types for which there are no encoding rules are transferred as a variable-length sequence of octets. The format of the data within the octet string is defined by the user.

5.2 Formal definition of data type objects

5.2.1 Data type class

The data type class specifies the root of the data type class tree. Its parent class "top" indicates the top of the FAL class tree.

FAL ASE: DATA TYPE ASE
CLASS: DATA TYPE

CLASS ID: Not used
PARENT CLASS: TOP
ATTRIBUTES:

1	(m)	Key Attribute:	Data type Name
2	(m)	Attribute:	Format (FIXED LENGTH, STRING, STRUCTURE, ARRAY)
3	(c)	Constraint:	Format = FIXED LENGTH | STRING
3.1	(m)	Attribute:	Octet Length
4	(c)	Constraint:	Format = STRUCTURE
4.1	(m)	Attribute:	Number of Fields
4.2	(m)	Attribute:	List of Fields
4.2.1	(o)	Attribute:	Field Name
4.2.2	(m)	Attribute:	Field Data type
5	(c)	Constraint:	Format = ARRAY
5.1	(m)	Attribute:	Number of Array Elements
5.2	(m)	Attribute :	Array Element Data type

5.2.2 Attributes

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

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

Number of Fields
This conditional attribute defines the number of fields in a structure. It is present when the value of the format attribute is "STRUCTURE".

List of Fields
This conditional attribute is an ordered list of fields contained in the structure. Each field is specified by its number and its type. Fields are numbered sequentially from 0 (zero) in the order in which they occur. Partial access to fields within a structure is not supported.

Field Name
This conditional, optional attribute specifies the name of the field. It may be present when the value of the format attribute is "STRUCTURE".

Field Data type
This conditional attribute specifies the data type of the field. It is present when the value of the format attribute is "STRUCTURE". This attribute may itself specify a constructed data type by referencing a constructed data type definition by embedding a constructed data type definition here.

Number of Array Elements
This conditional attribute defines the number of elements for the array type. Array elements are indexed starting at “0” through “n-1” where the size of the array is “n” elements. This attribute is present when the value of the format attribute is "ARRAY".

Array Element Data type
This conditional attribute specifies the data type for the elements of an array. All elements of the array have the same data type. It is present when the value of the format attribute is "ARRAY". This attribute may itself specify a constructed data type by referencing a constructed data type by its name.

5.3 FAL defined data types

5.3.1 Fixed length types

5.3.1.1 Integer8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Name	=	Integer8
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	1

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

5.3.1.2 Integer16

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Name	=	Integer16
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	2

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

5.3.1.3 Integer24

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Name	=	Integer24
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	3

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

5.3.1.4 Integer32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Name	=	Integer32
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	4

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

5.3.1.5 Unsigned8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Name	=	Unsigned8
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	1

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

5.3.1.6 Unsigned16

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Name	=	Unsigned16

2	Format	=	FIXED LENGTH
2.1	Octet Length	=	2

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

5.3.1.7 Unsigned24

CLASS: **Data type**

ATTRIBUTES:

1	Data type Name	=	Unsigned24
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	3

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

5.3.1.8 Unsigned32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Name	=	Unsigned32
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	4

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

5.3.1.9 Float32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Name	=	Float32
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	4

This type has a length of four octets. The format for Float32 is that defined by ANSI/IEEE 754 as single precision.

5.3.1.10 Float64

CLASS: **Data type**

ATTRIBUTES:

1	Data type Name	=	Float64
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	8

This type has a length of eight octets. The format for Float64 is that defined by ANSI/IEEE 754 as double precision.

5.3.1.11 Date

CLASS: **Data type**

ATTRIBUTES:

1	Data type Name	=	Date
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	3

This data type is consists of day, month and year minus 1900. This allows the representation of any date between 1 January 1900 and 31 December 2155.

5.3.1.12 Enumeration

CLASS:		**Data type**	
ATTRIBUTES:			
1	Data type Name	=	Enumeration
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	1

Data items that take on a single meaning from a list or table are encoded as Enumeration. This data type uses an unsigned integer of one octet length. The largest integer value is reserved and not used by any service.

5.3.1.13 Bit Field

CLASS:		**Data type**	
ATTRIBUTES:			
1	Data type Name	=	Bit Field
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	1

This data type is defined as a series of eight bits, numbered from 0 to 7. Communication of information encoded as single-bit data (such as status and diagnostic information) uses this data type.

5.3.1.14 Time

CLASS:		**Data type**	
ATTRIBUTES:			
1	Data type Name	=	Time
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	4

This data type is an unsigned binary integer and represents time in the increments of 1/32 of a millisecond. If this data type is used to represent time of day, then it indicates number of 1/32 of milliseconds since midnight.

5.3.1.15 Engineering unit

CLASS:		**Data type**	
ATTRIBUTES:			
1	Data type Name	=	Engineering unit
2	Format	=	FIXED LENGTH
2.1	Octet Length	=	1

This type defines the measurement unit of a measured variable. The interpretation of this data type is specified by the communication profile and beyond the scope of this part of the document.

5.3.2 String Types

5.3.2.1 Packed ASCII

CLASS:		**Data type**	
ATTRIBUTES:			
1	Data type Name	=	Packed ASCII
2	Format	=	STRING
2.1	Octet Length	=	1 to n

This type is a modified subset of the ASCII character code set. This subset is shown in Table 1.

Table 1 – Packed ASCII character set

Bits 4 and 5	**Bits 0 to 3**															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
2	SP[a]	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?

NOTE Most significant hexadecimal digit is top to bottom; least significant is left to right.

[a] SP indicates a space character.

5.3.2.2 ISO Latin-1

CLASS: **Data type**

ATTRIBUTES:

1	Data type Name	=	Latin-1
2	Format	=	STRING
2.1	Octet Length	=	1 to n

This type is a subset of the ISO/IEC 8859-1 (ISO Latin-1) string type. The subset is shown in Table 2.

Table 2 – ISO Latin-1 characters

Bits 4 to 7	**Bits 0 to 3**															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
2	SP[a]	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	B	c	D	e	f	g	h	i	j	k	l	m	n	o
7	p	q	R	s	T	u	v	w	x	y	z	{	|	}	~	▨
8	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
9	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
A	NBSP[b]	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY[c]	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	Ä	å	æ	ç	è	é	Ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	Ô	õ	ö	÷	ø	ù	Ú	û	ü	ý	þ	Ÿ

NOTE Most significant hexadecimal digit is top to bottom; least significant is left to right. Grayed out cells means that no character is assigned to this code.

[a] SP indicates a space character.

[b] NBSP indicates a non-breaking space character.

[c] SHY indicates a soft hyphen.

5.4 Data type ASE service specification

There are no operational services defined for the type object.

5.5 Summary of data types

This clause contains a summary of the defined data types as shown in Table 3.

Table 3 – Data type summary

Data type	Clause	Data type	Clause
Integer8	5.3.1.1	Float32	5.3.1.9
Integer16	5.3.1.2	Float64	5.3.1.10
Integre24	5.3.1.3	Date	5.3.1.11
Integer32	5.3.1.4	Enumeration	5.3.1.12
Unsigned8	5.3.1.5	Bit Field	5.3.1.13
Unsigned16	5.3.1.6	Time	5.3.1.15
Unsigned24	5.3.1.7	Engineering unit	5.3.1.15
Unsigned32	5.3.1.8	Packed ASCII	5.3.2.1
		ISO Latin-1	5.3.2.2

6 Communication model specification

6.1 Common parameters

Several parameters are used by more than one service. Instead of defining them with each service, the following common definitions are provided.

6.1.1 AREP ID

This parameter specifies sufficient information to identify the AREP of the remote end of the AR. One value of this parameter is reserved as broadcast address.

6.1.2 Response Code

If there is no communication error then this parameter specifies a command completion report indicating the status of the command's execution by the device. The possible values of this parameter are shown in Table 4. Its data type is Enumeration.

Table 4 – Response code values

Value	Description
Success	Command (read or Write) was executed properly.
Warning	Command (Write) was executed with the deviation as described in response (e.g., a value was set to its nearest legal value).
Error	Command (read or Write) was not executed properly. Response Code indicates the reason (e.g., the device is in Write Protect mode).

6.1.3 Application process Status

This parameter indicates the status provided by the FAL user and it is not associated with the completion of any command. Its length is one octet.

6.1.4 Extended status

This parameter indicates the VFD status. It is not associated with the completion of any command. Its length is one octet.

6.1.5 Preamble count

It specifies the minimum number of preamble octets in the DLPDU required by the destination DLE.

NOTE The value of this parameter can be obtained by using Application layer “Identify” service.

6.1.6 Communication status

This parameter specifies information about the communication failure. The possible values of this parameter are shown in Table 5.

Table 5 – Communication status values

Value	Description
Vertical parity Error	The parity of one or more of the octets received by the device was not odd.
Overrun error	At least one octet of data in the receive buffer of the PhE was overwritten before it was read (i.e., the receiver did not process incoming octet fast enough).
Framing error	The Stop bit of one or more octets received by the device was not detected by the PhE (i.e. a mark or 1 was not detected when it should have occurred).
Longitudinal parity error	The longitudinal parity calculated by the device did not match the Check octet at the end of the DLPDU.
Buffer overflow	The PhPDU or the DLPDU was too long for the receive buffer of the PhE or the DLE.
Device not available	The client did not receive any response from the server.

6.2 ASEs

6.2.1 Virtual field device ASE

6.2.1.1 Virtual field device class specification

The Virtual Field Device (VFD) is an abstract model for the description of the data and the behavior of an Application Process. VFDs contain APOs. The attributes of an APO are described by object descriptions. Services are defined for accessing a VFD's APOs, as shown in Figure 2.

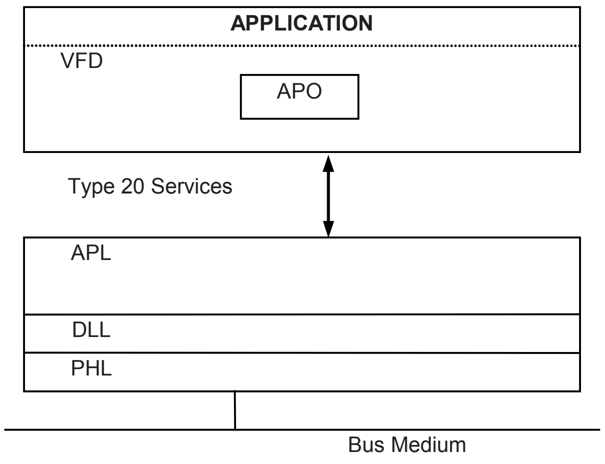

Figure 2 – VFD model

The services define no concrete interface for an implementation. They describe in an abstract form which functions may be used.

The application is not a subject of this document. It should only be indicated, how the abstractly described services may be made available to the application.

Only one VFD Object is present in a device.

6.2.1.2 Formal model

The VFD class specifies the attributes and services defined for application processes. Its parent class "top" indicates the top of the class tree.

ASE:	**VFD ASE**
CLASS:	**VFD**
CLASS ID:	**—**
PARENT CLASS:	**TOP**
1. (o) Key Attribute:	Not used
2. (m) Attribute:	Manufacturer ID
3. (m) Attribute:	Expanded Device Type
4. (m) Attribute:	Device ID
5. (m) Attribute:	Device Rev
6. (m) Attribute:	Soft Rev
7. (m) Attribute:	Hard Rev
8. (m) Attribute:	Phy Type
9. (m) Attribute:	Preamble Count
10. (m) Attribute:	Device Flag
11. (m) Attribute:	Command Rev
12. (m) Attribute:	Variable Count
13. (m) Attribute:	Config Change Counter
14. (m) Attribute:	Device ExtdStatus
15. (m) Attribute:	Distributor code
16. (m) Attribute:	Device profile
SERVICES:	
1. (m) Ops Service:	Identify

6.2.1.2.1 Attributes

Manufacturer ID

This attribute indicates the manufacturer that produced the device. A manufacturer is required to use the value assigned to it and is not permitted to use the value assigned to another manufacturer.

Expanded Device Type

This attribute indicates the manufacturer's type of the device i.e. the product name. The value of this attribute is assigned by the manufacturer. Its value specifies the set of commands and data objects supported by the device. The manufacturer is required to assigned unique value to each type of the device.

Device ID

This attribute indicates a serial number for the device. The manufacturer is required to assign a unique value for every device that has the identical values for Manufacturer ID and Device Type.

Device Rev

This attribute describes the revision level of the device. The value of this attribute is defined by the manufacturer. The value of this attribute describes the revision level of the set of commands and data objects supported by the device.

Soft Rev

This attribute indicates the revision level of the firmware in the device. The manufacturer is required to increment the value of this attribute for every new release of the device's firmware.

Hard Rev

This attribute indicates the revision level of the device hardware. The manufacturer is required to increment the value of this attribute for every major change of the device’s hardware. It is not necessary to track individual hardware component changes.

Phy Type

This attribute indicates the type of Physical layer signalling used by the device.

Preamble Count

This attribute indicates the minimum number of Preambles to be sent with the request message from the Master to the Slave device.

Device Flag

This attribute indicates other information about the device such as multi-sensor, non-volatile memory control, protocol bridge, etc.

Command Rev

This attribute indicates the major revision level of the Protocol supported by the device.

Variable Count

This attribute specifies the maximum number of objects (variables) that can be accessed from the device. The value of this attribute indicates the last Variable Code that a Client application can expect to be found in the device.

Configuration Change Counter

This attribute keeps track of number of device configuration changes. The device is required to increment the value of this attribute every time it receives a request to change the configuration using application layer services, or a user of the device changes the device configuration using local means such as local operator’s interface.

Device ExtdStatus

This attribute indicates the extended operational status of the device.

Distributor code

This attribute indicates the private label manufacturer that distributed the device.

Device profile

This attribute specifies the class to which the device belongs.

6.2.1.3 Identify service

6.2.1.3.1 Usage

This service is used to request the device identification information from the AP.

6.2.1.3.2 Service primitives

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

Table 6 – Identify service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP ID	M	M (=)		
Tag	C,U	M (=)		
Result (+)			S	S (=)
Manufacturer ID			M	M (=)
Expanded Device Type			M	M (=)
Device ID			M	M (=)
Device Rev			M	M (=)
Soft Rev			M	M (=)
Hard Rev			M	M (=)
Phy Type			M	M (=)
Preamble Count			M	M (=)
Device Flag			M	M (=)
Command Rev			M	M (=)
Variable Count			M	M (=)
Configuration Change Counter			M	M (=)
Device ExtdStatus			M	M (=)
Distributor code			M	M (=)
Device profile			M	M (=)
Response Code			M	M (=)
Application process Status			M	M (=)
Communication Error				S
Communication Status				M
Application process Status				C
No Match				S
Result (–)			S	S (=)
Response Code			M	M (=)
Application process Status			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter.				

Argument
The argument contains the parameters of the service request.

Tag
This parameter is used to identify the remote device. If the value of AREP ID is the Broadcast address, then this parameter is required, otherwise is is optional.

Result (+)
This selection type parameter indicates that the service request succeeded. The parameters for this selection are the attributes of VFD class object as described in 6.2.1.2.

Communication Error
This selection type parameter indicates that the service request failed due to communication error. The parameters for this selection provide the reason for the failure.

No Match
This selection is used when the indication has Tag parameter and it does not match with the Tag resident in the responding device.

Result (–)
This selection type parameter indicates that the service request failed. The parameters for this selection provide the reason for the failure.

6.2.1.3.3 Service procedure

The Confirmed Service Procedure specified in 3.5.3.3 applies to this service. If the indication primitive specifies a Tag and the responding device does not find the matching Tag then it does not send any response.

6.2.2 Variable ASE

6.2.2.1 Overview

The Variable ASE provides services to read or write a variable object in Server device. The variable is identified by a number. This numeric identifier completely describes the data type and structure of the object. The object can be any of the types defined in 5.3.

6.2.2.2 Variable model – common

6.2.2.2.1 Simple variable

6.2.2.2.1.1 Formal model

The simple variable object represents a single variable which is characterized by a defined Data type.

ASE:			**VARIABLE ASE**
CLASS:			**Simple VARIABLE**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(o)	Attribute:	Variable Name
3	(m)	Attribute:	Data type Name
SERVICES:			
1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
3	(o)	OpsService:	Information report

6.2.2.2.1.2 Attributes

Numeric Identifier
Identifies an instance of this object class.

Variable Name
It is the name assigned to the object.

Data type Name
It is the name assigned to the data type as specified in 5.3.1.

6.2.2.2.1.3 Services

Read
This service permits a client to read the value of a variable.

Write
This service permits a client to write the value of a variable.

Information report
This unconfirmed service permits a server to send value of a variable to any one or all clients.

6.2.2.2.2 Structure variable

6.2.2.2.2.1 Formal model

The structure object consists of a collection of simple variables of different data types. The record object is accessed completely and its individual fields can not be accessed separately.

ASE:			**VARIABLE ASE**
CLASS:			**Structure VARIABLE**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(o)	Attribute:	Variable Name
3	(m)	Attribute:	Number of Fields
4	(m)	Attribute:	List of Fields
4.1	(o)	Attribute:	Field Name
4.2	(m)	Attribute:	Field Data type Name
SERVICES:			
1	(o) OpsService:		Read
2	(o) OpsService:		Write
3	(o)	OpsService:	Information report

6.2.2.2.2.2 Attributes

Numeric Identifier
Identifies an instance of this object class.

Variable Name
It is the name assigned to the object.

Number of Fields
This attribute defines the number of fields in a structure.

Field Name
This optional attribute specifies the name of the field.

Field Data type Name
It is the name assigned to the data type as specified in 5.3.1.

6.2.2.2.2.3 Services

Read
This service permits a client to read the value of a variable.

Write
This service permits a client to read the value of a variable.

Information report
This unconfirmed service permits a server to send value of a variable to any one or all clients.

6.2.2.2.3 Array of simple variable

6.2.2.2.3.1 Formal model

This Array Object is used to define a constructed variable in which all elements have the identical Data type and length and are of Simple Variable class.

ASE:			**VARIABLE ASE**
CLASS:			**Array of Simple VARIABLE**
PARENT CLASS:			**Simple VARIABLE**
ATTRIBUTES:			
4	(m)	Attribute:	Array size

6.2.2.2.3.2 Attributes

Array size
It states how many elements are contained in the Array. If the value of this attribute is “Variable” then the number of elements is not fixed.

6.2.2.3 Variable ASE service specification

6.2.2.3.1 Supported services

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

– Read variable from Server device,
– Write variable to Server device, and
– Send Information report from a server device.

There are separate services for each simple or record variable, which is identified by a number. The format of the services is common to all variables. The common format of Read service is defined in 6.2.2.3.2. The common format of Write service is defined in 6.2.2.3.4. The common format of Information report service is defined in 6.2.2.3.5. The format of each variable is specified in the individual service to read or write that variable.

NOTE The service definitions in this subclause are not independent of the variable being accessed. This subclause only shows the format of the service. The service specific definitions include the variable model and the service specific parameters.

6.2.2.3.2 Primitive correlation

At the client, there is only one outstanding request for one value of AREP ID. The AREP ID is conveyed as destination address in the data link layer service primitives. Therefore, the confirm primitive is correlated with the request primitive using AREP ID. At the server, there is only one outstanding indication primitive at any time. Therefore, the response primitive is correlated with the indication primitive waiting for the response.

6.2.2.3.3 Read service – common format

6.2.2.3.3.1 Service overview

This service permits a client to read the value of a variable from the server.

6.2.2.3.3.2 Service primitives

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

Table 7 – Read service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP ID	M	M (=)		
Numeric Identifier	M	M (=)	M (=)	M (=)
Additional identifier	C	M (=)	M (=)	M (=)
Subindex	C	M (=)	M (=)	M (=)
Preamble count	M			
Result (+)			S	S (=)
Variable value			M	M (=)
Response Code			M	M (=)
Application process Status			M	M (=)
Communication Error				S
Communication Status				M
Application process Status				C
Result (–)			S	S (=)
Response Code			M	M (=)
Application process Status			M	M (=)

Argument
The argument contains the parameters of the service request.

Numeric Identifier
This parameter identifies the variable. This is the Numeric Identifier assigned to the variable to be read.

Additional identifier
For some of the Numeric Identifiers, this parameter provides additional information required to identify the variable(s) to be read.

Subindex
This parameter identifies the individual element in an array variable by its position within the variable. This can be either a numeric value or an Enumeration.

Variable value
This parameter specifies the value of the variable read from the server.

Communication status
If this status is ‘Device not available’ then the confirm primitive does not return ‘Application process status’.

6.2.2.3.4 Write service – common format

6.2.2.3.4.1 Service overview

This service permits a client to write the value of a variable to the server.

6.2.2.3.4.2 Service primitives

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

Table 8 – Write service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP ID	M	M (=)		
Numeric Identifier	M	M (=)	M	M (=)
Additional identifier	C	M (=)	M	M (=)
Subindex	C	M (=)	M	M (=)
Preamble count	M			
Variable value	M	M (=)		
Result (+)			S	S (=)
Variable value			M	M (=)
Response Code			M	M (=)
Application process Status			M	M (=)
Communication Error				S
Communication Status				M
Application process Status				C
Result (–)			S	S (=)
Response Code			M	M (=)
Application process Status			M	M (=)

Argument
The argument contains the parameters of the service request.

Numeric Identifier
This parameter identifies the variable. This is the key attribute assigned to the variable to be written. It is also called command number.

Additional identifier
For some of the Numeric Identifiers, this parameter provides additional information required to identify the variable(s) to be written.

Subindex
This parameter identifies the individual element in an array variable by its position within the variable. This can be either a numeric value or an Enumeration.

Variable value
In the request primitive, this parameter specifies the desired value of the variable to be written. In the response primitive, this parameter states the actual value of the variable that was written. In some cases, the responding server writes a value different than the value in indication primitive. For example, it may round the value to the nearest allowed value or the nearest value the server can support.

NOTE The returned variable value can be used by the FAL-user to validate the write operation.

Communication status
If this status is ‘Device not available’ then the confirm primitive does not return ‘Application process status’.

6.2.2.3.5 Information report service – common format

6.2.2.3.5.1 Service overview

This service permits a slave to send the value of a variable and master to receive it without using a request from the master. It is used to publish variable data on a cyclic basis.

6.2.2.3.5.2 Service primitives

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

Table 9 – Information report parameters

Parameter name	Req	Ind
Argument		
AREP ID	M	M (=)
Numeric Identifier	M	M (=)
Additional identifier	C	M (=)
Subindex	C	M (=)
Variable value	M	M (=)
Response Code	M	M (=)
Application process status	M	M (=)
Extended status	C	M(=)
Variable value	M	M (=)

Argument
The argument contains the parameters of the service request.

Numeric Identifier
This parameter identifies the variable. This is the key attribute assigned to the variable to be read. It is also called command number.

Additional identifier
For some of the Numeric Identifiers, this parameter provides additional information required to identify the variable(s) to be read.

Subindex
This parameter identifies the individual element in an array variable by its position within the variable. This can be either a numeric value or an Enumeration.

Variable value
This parameter specifies the value reported by the server.

6.2.3 Action ASE

6.2.3.1 Service overview

This service permits a client to command a specified action of the server. The common format of Action service is defined in this clause. For every action, there is a separate service.

NOTE The service definitions in this subclause are not independent of the action being requested. This subclause only shows the format of the service. The service specific definitions include the action behavior and the service specific parameters.

6.2.3.2 Primitive correlation

At the client, there is only one outstanding request for one value of AREP ID. The AREP ID is conveyed as destination address in the transport layer service primitives. Therefore, the confirm primitive is correlated with the request primitive using AREP ID. At the server, there is only one outstanding indication primitive at any time. Therefore, the response primitive is correlated with the indication primitive waiting for the response.

6.2.3.3 Action model

6.2.3.3.1 General

Some of the actions require a variable to validate the action. The model for such actions is defined in this subclause.

6.2.3.3.2 Formal model

ASE:			**Simple Action**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier

ASE:			**Qualified Action**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Qualifier Name
3	(m)	Attribute:	Data type Name

ASE:			**Simple Variable Action**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Variable Name
3	(m)	Attribute:	Data type Name

ASE:			**Structure Variable Action**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Variable Name
3	(m)	Attribute:	Number of Fields
4	(m)	Attribute:	List of Fields
4.1	(o)	Attribute:	Field Name
4.2	(m)	Attribute:	Field Data type Name

6.2.3.3.3 Attributes

See 6.2.2.2 for the definition of attributes.

6.2.3.4 Action service primitives – common format

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

Table 10 – Action service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP ID	M	M (=)		
Numeric Identifier	M	M (=)	M	M (=)
Additional identifier	C	M (=)	M	M (=)
Subindex	C	M (=)	M	M (=)
Preamble count	M			
Variable value	C	M (=)		
Result (+)			S	S (=)
Variable value			S	S (=)
Response Code			M	M (=)
Application process Status			M	M (=)
Communication Error				S
Communication Status				M
Application process Status				C
Result (–)			S	S (=)
Response Code			M	M (=)
Application process Status			M	M (=)

Argument

The argument contains the parameters of the service request.

Numeric Identifier

This parameter identifies the action. It is also called command number.

Action qualifier

For some of the Numeric Identifiers, this parameter provides additional information required to identify action to be performed. If the action requires a variable and if the Numeric identifier is not sufficient, then this parameter identifies that variable.

Subindex

If the action requires an array variable, then this parameter identifies the individual element in that array variable. This parameter can be either a numeric or an enumeration.

NOTE 1 This parameter is present if the action is to add an element in the array or delete an element from the array.

Variable value

If the action requires a variable, then this parameter specifies the value of the variable. If the action is “add” then in the request parameter, this parameter specifies the desired value of the variable to be added; in the response primitive, this parameter states the actual value of the variable that was added.

NOTE 2 The returned variable value can be used by the FAL-user to validate the write operation.

Communication status

If this status is ‘Device not available’ then the confirm primitive does not return ‘Application process status’.

6.2.4 Device application services

6.2.4.1 Read primary variable

6.2.4.1.1 Service overview

This service permits a client to read the value of Primary variable and its Unit from the server.

6.2.4.1.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 1
2	Variable Sub Index	Not present
3	Variable Name	= Primary variable
4	Number of Fields	= 2
5	List of Fields	
5.1	Field Name	= Primary variable Unit
5.2	Field Data type Name	= Engineering unit
5.3	Field Name	= Primary variable
5.4	Field Data type Name	= Float32

6.2.4.2 Read loop current and percent of range

6.2.4.2.1 Service overview

This service permits a client to read the value of the Loop current and its associated Percent of range from the server.

6.2.4.2.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 2
2	Variable Sub Index	Not present
3	Variable Name	= Loop current
4	Number of Fields	= 2
5	List of Fields	
5.1	Field Name	= Loop current
5.2	Field Data type Name	= Float32
5.3	Field Name	= Percent of range
5.4	Field Data type Name	= Float32

6.2.4.3 Read dynamic variables and loop current

6.2.4.3.1 Service overview

This service permits a client to read the value of the Loop current and up to four predefined Dynamic variables from the server. The number of variables in the response data depends upon the type of the device.

6.2.4.3.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 3
2	Variable Sub Index	Not present

3		Variable Name	= Dynamic variables
4		List of Fields	
4.1	(c)	Constraint	Number of fields = {3 | 5 | 7 | 9}
4.1.1	(m)	Field Name	= Loop current
4.1.2	(m)	Field Data type Name	= Float32
4.1.3	(m)	Field Name	= Primary variable Unit
4.1.4	(m)	Field Data type Name	= Engineering unit
4.1.5	(m)	Field Name	= Primary variable
4.1.6	(m)	Field Data type Name	= Float32
4.2	(c)	Constraint	Number of Fields = {5 | 7 | 9}
4.2.1	(m)	Field Name	= Secondary variable Unit
4.2.2	(m)	Field Data type Name	= Engineering unit
4.2.3	(m)	Field Name	= Secondary variable
4.2.4	(m)	Field Data type Name	= Float32
4.3	(c)	Constraint	Number of Fields = {7 | 9}
4.3.1	(m)	Field Name	= Tertiary variable Unit
4.3.2	(m)	Field Data type Name	= Engineering unit
4.3.3	(m)	Field Name	= Tertiary variable
4.3.4	(m)	Field Data type Name	= Float32
4.4	(c)	Constraint	Number of Fields = 9
4.4.1	(m)	Field Name	= Quaternary variable Unit
4.4.2	(m)	Field Data type Name	= Engineering unit
4.4.3	(m)	Field Name	= Quaternary variable
4.4.4	(m)	Field Data type Name	= Float32

The number of fields can be either 3 or 5 or 7 or 9. The fields marked as 4.1.1 to 4.1.6 are always present. If the number of fields is 5 or 7 or 9, then fields 4.2.1 to 4.2.4 are present. If the number of fields is 7 or 9, then fields 4.3.1 to 4.3.4 are present. If the number of fields is 9, then fields 4.4.1 to 4.4.4 are present.

6.2.4.4 Write loop configuration

6.2.4.4.1 Service overview

This service permits a client to write the value of the Polling address and the Loop current mode to the server.

6.2.4.4.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 6
2	Variable Sub Index	Not present
3	Variable Name	= Loop configuration
4	Number of Fields	= 2
5	List of Fields	
5.1	Field Name	= Polling address
5.2	Field Data type Name	= Unsigned8
5.3	Field Name	= Loop current mode
5.4	Field Data type Name	= Enumeration

6.2.4.5 Read loop configuration

6.2.4.5.1 Service overview

This service permits a client to read the value of the Polling address and the Loop current mode from the server.

6.2.4.5.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 7
2	Variable Sub Index	Not present
3	Variable Name	= Loop configuration
4	Number of Fields	= 2
5	List of Fields	
5.1	Field Name	= Polling address
5.2	Field Data type Name	= Unsigned8
5.3	Field Name	= Loop current mode
5.4	Field Data type Name	= Enumeration

6.2.4.6 Read dynamic variable families classifications

6.2.4.6.1 Service overview

This service permits a client to read the value of the Classification associated with the Dynamic variables.

6.2.4.6.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 8
2	Variable Sub Index	Not present
3	Variable Name	= Dynamic variables classification
4	Number of Fields	= 4
5	List of Fields	
5.1	Field Name	= Primary variable classification
5.2	Field Data type Name	= Enumeration
5.3	Field Name	= Secondary variable classification
5.4	Field Data type Name	= Enumeration
5.5	Field Name	= Tertiary variable classification
5.6	Field Data type Name	= Enumeration
5.7	Field Name	= Quaternary variable classification
5.8	Field Data type Name	= Enumeration

6.2.4.7 Read device variables with status

6.2.4.7.1 Service overview

This service permits a client to read the value of up to eight Device or Dynamic variable from the server. Client may request any number of Device variables. The variables are identified by assigning the Variable code to Variable sub index. The number of variables in the response data matches the number of variables in the request.

6.2.4.7.2 Request Variable Sub Index model

CLASS: **STRUCTURE VARIABLE**
ATTRIBUTES:

1 Numeric Identifier = 9
2 Variable Name = Device Variables
3 (c) Constraint Variable Sub Index = up to 8
3.1 (m) Sub Index 1
3.1.1 (m) Field Name = Variable code
3.1.2 (m) Field Data type Name = Unsignde8
3.2 (c) Sub Index 2
3.2.1 (m) Field Name = Variable code
3.2.2 (m) Field Data type Name = Unsignde8
3.3 (c) Sub Index 3
3.3.1 (m) Field Name = Variable code
3.3.2 (m) Field Data type Name = Unsignde8
3.4 (c) Sub Index 4
3.4.1 (m) Field Name = Variable code
3.4.2 (m) Field Data type Name = Unsignde8
3.5 (c) Sub Index 5
3.5.1 (m) Field Name = Variable code
3.5.2 (m) Field Data type Name = Unsignde8
3.6 (c) Sub Index 6
3.6.1 (m) Field Name = Variable code
3.6.2 (m) Field Data type Name = Unsignde8
3.7 (c) Sub Index 7
3.7.1 (m) Field Name = Variable code
3.7.2 (m) Field Data type Name = Unsignde8
3.8 (c) Sub Index 8
3.8.1 (m) Field Name = Variable code
3.8.2 (m) Field Data type Name = Unsignde8

6.2.4.7.3 Response Variable model

CLASS: **STRUCTURE VARIABLE**
ATTRIBUTES:

1 Numeric Identifier = 9
2 Variable Name = Device Variables
3 List of Fields
3.1 (c) Constraint Number of Variables = {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8}
3.1.1 (m) Field Name = Device ExtdStatus
3.1.2 (m) Field Data type Name = Bit Field
3.1.3 (m) Field Name = Variable code
3.1.4 (m) Field Data type Name = Unsigned8
3.1.5 (m) Field Name = Variable class
3.1.6 (m) Field Data type Name = Enumeration
3.1.7 (m) Field Name = Variable Unit
3.1.8 (m) Field Data type Name = Engineering unit
3.1.9 (m) Field Name = Variable
3.1.10 (m) Field Data type Name = Float32
3.1.11 (m) Field Name = Variable status
3.1.12 (m) Field Data type Name = Bit Field
3.2 (c) Constraint Number of Variables = {2 | 3 | 4 | 5 | 6 | 7 | 8}
3.2.1 (m) Field Name = Variable code

3.2.2	(m)	Field Data type Name	= Unsigned8
3.2.3	(m)	Field Name	= Variable class
3.2.4	(m)	Field Data type Name	= Enumeration
3.2.5	(m)	Field Name	= Variable Unit
3.2.6	(m)	Field Data type Name	= Engineering unit
3.2.7	(m)	Field Name	= Variable
3.2.8	(m)	Field Data type Name	= Float32
3.2.9	(m)	Field Name	= Variable status
3.2.10	(m)	Field Data type Name	= Bit Field
3.3	(c)	Constraint	Number of Variables = {3 | 4 | 5 | 6 | 7 | 8}
3.3.1	(m)	Field Name	= Variable code
3.3.2	(m)	Field Data type Name	= Unsigned8
3.3.3	(m)	Field Name	= Variable class
3.3.4	(m)	Field Data type Name	= Enumeration
3.3.5	(m)	Field Name	= Variable Unit
3.3.6	(m)	Field Data type Name	= Engineering unit
3.3.7	(m)	Field Name	= Variable
3.3.8	(m)	Field Data type Name	= Float32
3.3.9	(m)	Field Name	= Variable status
3.3.10	(m)	Field Data type Name	= Bit Field
3.4	(c)	Constraint	Number of Variables = {4 | 5 | 6 | 7 | 8}
3.4.1	(m)	Field Name	= Variable code
3.4.2	(m)	Field Data type Name	= Unsigned8
3.4.3	(m)	Field Name	= Variable class
3.4.4	(m)	Field Data type Name	= Enumeration
3.4.5	(m)	Field Name	= Variable Unit
3.4.6	(m)	Field Data type Name	= Engineering unit
3.4.7	(m)	Field Name	= Variable
3.4.8	(m)	Field Data type Name	= Float32
3.4.9	(m)	Field Name	= Variable status
3.4.10	(m)	Field Data type Name	= Bit Field
3.5	(c)	Constraint	Number of Variables = {5 | 6 | 7 | 8}
3.5.1	(m)	Field Name	= Variable code
3.5.2	(m)	Field Data type Name	= Unsigned8
3.5.3	(m)	Field Name	= Variable class
3.5.4	(m)	Field Data type Name	= Enumeration
3.5.5	(m)	Field Name	= Variable Unit
3.5.6	(m)	Field Data type Name	= Engineering unit
3.5.7	(m)	Field Name	= Variable
3.5.8	(m)	Field Data type Name	= Float32
3.5.9	(m)	Field Name	= Variable status
3.5.10	(m)	Field Data type Name	= Bit Field
3.6	(c)	Constraint	Number of Variables = {6 | 7 | 8}
3.6.1	(m)	Field Name	= Variable code
3.6.2	(m)	Field Data type Name	= Unsigned8
3.6.3	(m)	Field Name	= Variable class
3.6.4	(m)	Field Data type Name	= Enumeration
3.6.5	(m)	Field Name	= Variable Unit
3.6.6	(m)	Field Data type Name	= Engineering unit
3.6.7	(m)	Field Name	= Variable
3.6.8	(m)	Field Data type Name	= Float32
3.6.9	(m)	Field Name	= Variable status
3.6.10	(m)	Field Data type Name	= Bit Field

3.7	(c)	Constraint	Number of Variables = {7 | 8}
3.7.1	(m)	Field Name	= Variable code
3.7.2	(m)	Field Data type Name	= Unsigned8
3.7.3	(m)	Field Name	= Variable class
3.7.4	(m)	Field Data type Name	= Enumeration
3.7.5	(m)	Field Name	= Variable Unit
3.7.6	(m)	Field Data type Name	= Engineering unit
3.7.7	(m)	Field Name	= Variable
3.7.8	(m)	Field Data type Name	= Float32
3.7.9	(m)	Field Name	= Variable status
3.7.10	(m)	Field Data type Name	= Bit Field
3.8	(c)	Constraint	Number of Variables = 8
3.8.1	(m)	Field Name	= Variable code
3.8.2	(m)	Field Data type Name	= Unsigned8
3.8.3	(m)	Field Name	= Variable class
3.8.4	(m)	Field Data type Name	= Enumeration
3.8.5	(m)	Field Name	= Variable Unit
3.8.6	(m)	Field Data type Name	= Engineering unit
3.8.7	(m)	Field Name	= Variable
3.8.8	(m)	Field Data type Name	= Float32
3.8.9	(m)	Field Name	= Variable status
3.8.10	(m)	Field Data type Name	= Bit Field
3.9	(m)	Field Name	= Slot 0 time stamp
3.10	(m)	Field Data type Name	= Time

The number of variables can be any number from 1 to 8. The fields marked as 3.1.1 to 3.1.12 and 3.9 to 3.10 are always present.

6.2.4.8 Read message

6.2.4.8.1 Service overview

This service permits a client to read the value of the Message string from the server.

6.2.4.8.2 Variable model

CLASS: **Simple VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 12
2	Variable Sub Index	Not present
3	Variable Name	= Message
4	Field Name	= Message
5	Field Data type Name	= Packed ASCII

6.2.4.9 Read tag, descriptor and date

6.2.4.9.1 Service overview

This service permits a client to read the value of the Tag, Descriptor and Date from the server.

6.2.4.9.2 Variable model

CLASS: **STRUCTURE VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 13

2	Variable Sub Index	Not present
3	Variable Name	= Tag Descriptor
4	Number of Fields	= 3
5	List of Fields	
5.1	Field Name	= Tag
5.2	Field Data type Name	= Packed ASCII
5.3	Field Name	= Descriptor
5.4	Field Data type Name	= Packed ASCII
5.5	Field Name	= Date
5.6	Field Data type Name	= Date

6.2.4.10 Read primary variable transducer information

6.2.4.10.1 Service overview

This service permits a client to read the value of the Transducer serial number, Limits/Minimum span Unit, Upper transducer limit, Lower transducer limit, and Minimum span for the Primary variable transducer from the server.

6.2.4.10.2 Variable model

CLASS: **STRUCTURE VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 14
2	Variable Sub Index	Not present
3	Variable Name	= Primary variable transducer Information
4	Number of Fields	= 5
5	List of Fields	
5.1	Field Name	= Transducer serial number
5.2	Field Data type Name	= Unsigned24
5.3	Field Name	= Transducer limits and minimum span Units
5.4	Field Data type Name	= Engineering unit
5.5	Field Name	= Upper transducer limit
5.6	Field Data type Name	= Float32
5.7	Field Name	= Lower transducer limit
5.8	Field Data type Name	= Float32
5.9	Field Name	= Minimum span
5.10	Field Data type Name	= Float32

6.2.4.11 Read device information

6.2.4.11.1 Service overview

This service permits a client to read the value of the PV alarm selection code, PV transfer function, PV upper and lower range values Unit, PV upper range, PV lower range, PV damping, Write protect, Private label distributor and PV analog channel from the server.

6.2.4.11.2 Variable model

CLASS: **STRUCTURE VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 15
2	Variable Sub Index	Not present
3	Variable Name	= Device information
4	Number of Fields	= 9
5	List of Fields	

5.1	Field Name	= PV alarm selection
5.2	Field Data type Name	= Enumeration
5.3	Field Name	= PV transfer function
5.4	Field Data type Name	= Enumeration
5.5	Field Name	= PV Range Unit
5.6	Field Data type Name	= Engineering unit
5.7	Field Name	= PV upper range
5.8	Field Data type Name	= Float32
5.9	Field Name	= PV lower range
5.10	Field Data type Name	= Float32
5.11	Field Name	= PV damping
5.12	Field Data type Name	= Float32
5.13	Field Name	= Write protect
5.14	Field Data type Name	= Enumeration
5.15	Field Name	= Reserved
5.16	Field Data type Name	= Enumeration
5.17	Field Name	= PV analog channel
5.18	Field Data type Name	= Bit Field

6.2.4.12 Read final assembly number

6.2.4.12.1 Service overview

This service permits a client to read the value of the Final assembly number associated with the device from the server.

6.2.4.12.2 Variable model

CLASS: **Simple VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 16
2	Variable Sub Index	Not present
3	Variable Name	= Final assembly number
4	Field Name	= Final assembly number
5	Field Data type Name	= Unsigned24

6.2.4.13 Write message

6.2.4.13.1 Service overview

This service permits a client to write the value of the Message string to the server.

6.2.4.13.2 Variable model

CLASS: **Simple VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 17
2	Variable Sub Index	Not present
3	Variable Name	= Message
4	Field Name	= Message
5	Field Data type Name	= Packed ASCII

6.2.4.14 Write tag, descriptor and date

6.2.4.14.1 Service overview

This service permits a client to write the value of the Tag, Descriptor and Date to the server.

6.2.4.14.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 18
2	Variable Sub Index	Not present
3	Variable Name	= Tag Descriptor
4	Number of Fields	= 3
5	List of Fields	
5.1	Field Name	= Tag
5.2	Field Data type Name	= Packed ASCII
5.3	Field Name	= Descriptor
5.4	Field Data type Name	= Packed ASCII
5.5	Field Name	= Date
5.6	Field Data type Name	= Date

6.2.4.15 Write final assembly number

6.2.4.15.1 Service overview

This service permits a client to write the value of the Final assembly number associated with the device to the server.

6.2.4.15.2 Variable model

CLASS:		**Simple VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 19
2	Variable Sub Index	Not present
3	Variable Name	= Final assembly number
4	Field Name	= Final assembly number
5	Field Data type Name	= Unsigned24

6.2.4.16 Read long tag

6.2.4.16.1 Service overview

This service permits a client to read the value of the Long tag from the server.

6.2.4.16.2 Variable model

CLASS:		**Simple VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 20
2	Variable Sub Index	Not present
3	Variable Name	= Long tag
4	Field Name	= Long tag
5	Field Data type Name	= Latin-1

6.2.4.17 Write long tag

6.2.4.17.1 Service overview

This service permits a client to write the value of the Long tag from the server.

6.2.4.17.2 Variable model

CLASS:		**Simple VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 22
2	Variable Sub Index	Not present
3	Variable Name	= Long tag
4	Field Name	= Long tag
5	Field Data type Name	= Latin-1

6.2.4.18 Reset configuration changed flag

This service is specified in IEC 62591:2010, 7.3.2.4.18.

6.2.4.19 Perform self test

This service is specified in IEC 62591:2010, 7.3.2.4.19.

6.2.4.20 Perform device reset

This service is specified in IEC 62591:2010, 7.3.2.4.20.

6.2.4.21 Read additional device status

This service is specified in IEC 62591:2010, 7.3.2.4.21.

6.2.4.22 Reset more status available

This service is specified in IEC 62591:2010, 7.3.2.4.22.

6.2.4.23 Read device variable information

This service is specified in IEC 62591:2010, 7.3.2.4.23.

6.2.4.24 Write device variable

This service is specified in IEC 62591:2010, 7.3.2.4.24.

6.2.4.25 Write primary variable range values

6.2.4.25.1 Service overview

This service sets the relationship between the loop current 4,00 mA, 20,0 mA points and the primary variable value. The upper range value of the primary variable is independent of the lower range value and the upper range value can be smaller than the lower range value.

6.2.4.25.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 35
2	Variable Name	= Primary variable range

3	Number of Fields	= 3
4	List of Fields	
4.1	Field Name	= Range unit
4.2	Field Data type Name	= Engineering unit
4.3	Field Name	= Upper range
4.4	Field Data type Name	= Float32
4.5	Field Name	= Lower range
4.6	Field Data type Name	= Float32

6.2.4.26 Enter-exit fixed current mode

6.2.4.26.1 Service overview

This service is used to put the device in fixed Loop current mode or exit from the fixed Loop current mode.

6.2.4.26.2 Request variable model

CLASS: **Simple VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 40
2	Variable Name	= Fixed current level
3	Variable Data type Name	= Float32

6.2.4.26.3 Response variable model

CLASS: **Simple VARIABLE**

ATTRIBUTES:

4	Numeric Identifier	= 40
5	Variable Name	= Actual current level
6	Variable Data type Name	= Float32

6.2.4.27 Write primary variable unit

6.2.4.27.1 Service overview

This service is used to set the measurement unit for the Primary variable.

6.2.4.27.2 Variable model

CLASS: **Simple VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 44
2	Variable Name	= Primary variable unit code
3	Variable Data type Name	= Engineering unit

6.2.4.28 Trim loop current zero

6.2.4.28.1 Service overview

This service is used to trim the internal zero calibration, so that the Loop current matches the value measured by the Master.

6.2.4.28.2 Variable model

CLASS:	**Simple VARIABLE**
ATTRIBUTES:	
1 Numeric Identifier	= 45
2 Variable Name	= Loop current level
3 Variable Data type Name	= Float32

6.2.4.29 Trim loop current gain

6.2.4.29.1 Service overview

This service is used to trim the internal gain calibration, so that the Loop current matches the value measured by the Master.

6.2.4.29.2 Variable model

CLASS:	**Simple VARIABLE**
ATTRIBUTES:	
1 Numeric Identifier	= 46
2 Variable Name	= Loop current level
3 Variable Data type Name	= Float32

6.2.4.30 Read dynamic variable assignment

6.2.4.30.1 Service overview

This service reads the Device variable numbers that are assigned to the Primary, Secondary, Tertiary and Quaternary variables.

6.2.4.30.2 Variable model

CLASS:	**STRUCTURE VARIABLE**
ATTRIBUTES:	
1 Numeric Identifier	= 50
2 Variable Name	= Dynamic variable assignment
3 Number of Fields	= 4
4 List of Fields	
4.1 Field Name	= Primary variable assignment
4.2 Field Data type Name	= Unsigned8
4.3 Field Name	= Secondary variable assignment
4.4 Field Data type Name	= Unsigned8
4.5 Field Name	= Tertiary variable assignment
4.6 Field Data type Name	= Unsigned8
4.7 Field Name	= Quaternary variable assignment
4.8 Field Data type Name	= Unsigned8

6.2.4.31 Write dynamic variable assignment

6.2.4.31.1 Service overview

This service writes the Device variable numbers that are assigned to the Primary, Secondary, Tertiary and Quaternary variables.

6.2.4.31.2 Variable model

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 51
2	Variable Name	= Dynamic variable assignment
3	Number of Fields	= 4
4	List of Fields	
4.1	Field Name	= Primary variable assignment
4.2	Field Data type Name	= Unsigned8
4.3	Field Name	= Secondary variable assignment
4.4	Field Data type Name	= Unsigned8
4.5	Field Name	= Tertiary variable assignment
4.6	Field Data type Name	= Unsigned8
4.7	Field Name	= Quaternary variable assignment
4.8	Field Data type Name	= Unsigned8

6.2.4.32 Write number of response preambles

6.2.4.32.1 Service overview

This service is used to set the number of preamble characters to be sent by a Slave device before the start of a response PhPDU.

6.2.4.32.2 Variable model

CLASS:		**Simple VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 59
2	Variable Name	= Preamble count
3	Variable Data type Name	= Unsigned8

6.2.4.33 Read device variable trim points

6.2.4.33.1 Service overview

This service reads the last successful trim points of the Device variable.

6.2.4.33.2 Request and response parameters

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 80
2	Subindex	
2.1	Subindex Name	= Variable code
2.2	Subindex Data type	= Unsigned8

6.2.4.33.3 Variable model

CLASS:		**Array of Structure VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 80
2	Variable Name	= Device variable trim points record
3	Number of Fields	= 3
4	List of Fields	
4.1	Field Name	= Trim point unit

4.2	Field Data type Name	= Engineering unit
4.3	Field Name	= Lower trim point
4.4	Field Data type Name	= Float32
4.5	Field Name	= Upper trim point
4.6	Field Data type Name	= Float32
5	Array size	= Variable

6.2.4.34 Read device variable trim guidelines

6.2.4.34.1 Service overview

This service reads the information that a Host will need to guide a user through a correct selection of trim points.

6.2.4.34.2 Request and response parameters

CLASS: **STRUCTURE VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 81
2	Subindex	
2.1	Subindex Name	= Variable code
2.2	Subindex Data type	= Unsigned8

6.2.4.34.3 Variable model

CLASS: **Array of Structure VARIABLE**

ATTRIBUTES:

1	Numeric Identifier	= 81
2	Variable Name	= Device variable trim guidelines record
3	Number of Fields	= 7
4	List of Fields	
4.1	Field Name	= Trim points supported
4.2	Field Data type Name	= Enumeration
4.3	Field Name	= Trim point unit
4.4	Field Data type Name	= Engineering unit
4.5	Field Name	= Minimum lower trim point
4.6	Field Data type Name	= Float32
4.7	Field Name	= Maximum lower trim point
4.8	Field Data type Name	= Float32
4.9	Field Name	= Minimum upper trim point
4.10	Field Data type Name	= Float32
4.11	Field Name	= Maximum upper trim point
4.12	Field Data type Name	= Float32
4.13	Field Name	= Minimum trim point difference
4.14	Field Data type Name	= Float32
5	Array size	= Variable

6.2.4.35 Write device variable trim point

6.2.4.35.1 Service overview

This service writes the value of one trim point for a Device variable to perform a calibration adjustment.

6.2.4.35.2 Request and response parameters

CLASS:		**STRUCTURE VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 82
2	Subindex	
2.1	Subindex Name	= Variable code
2.2	Subindex Data type	= Unsigned8

6.2.4.35.3 Variable model

CLASS:		**Array of Structure VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 82
2	Variable Name	= Device variable trim point record
3	Number of Fields	= 3
4	List of Fields	
4.1	Field Name	= Trim points type
4.2	Field Data type Name	= Enumeration
4.3	Field Name	= Trim point unit
4.4	Field Data type Name	= Engineering unit
4.5	Field Name	= Trim point value
4.6	Field Data type Name	= Float32
5	Array size	= Variable

6.2.4.36 Reset device variable trim

6.2.4.36.1 Service overview

This service resets the value of trim points for a Device variable to the default factory values.

6.2.4.36.2 Request and response parameters

CLASS:		**Simple VARIABLE**
ATTRIBUTES:		
1	Numeric Identifier	= 83
2	Variable Name	= Variable code
3	Variable Data type Name	= Unsigned8

6.2.4.37 Write publish data period

This service is specified in IEC 62591:2010, 7.3.2.4.26.

6.2.4.38 Write publish data trigger

This service is specified in IEC 62591:2010, 7.3.2.4.27.

6.2.4.39 Read publish data mode configuration

This service is specified in IEC 62591:2010, 7.3.2.4.28.

6.2.4.40 Flush delayed responses

This service is specified in IEC 62591:2010, 7.3.2.4.29.

6.2.4.41 Write publish data device variables

This service is specified in IEC 62591:2010, 7.3.2.4.30.

6.2.4.42 Write publish data mode command number

This service is specified in IEC 62591:2010, 7.3.2.4.31.

6.2.4.43 Write publish data mode control

This service is specified in IEC 62591:2010, 7.3.2.4.32.

6.3 ARs

6.3.1 Application relationship ASE

6.3.1.1 Overview

ARs are composed of a set of endpoints of compatible classes. One endpoint of each AR has to be Master class and the other end has to be Slave class. Each device can have only one instance of an Slave class AREP.

A device can use more than one Master class AREP to communicate with several Slave class AREPs. But, only one AR is active at any time. The user at Master class AREP provides the identification of the Slave class AREP as a parameter in the request primitive.

6.3.1.2 Master class AREP

6.3.1.2.1 Formal model

ASE:			**AR ASE**
CLASS:			**Master**
CLASS ID:			**Not used**
PARENT CLASS:			**Top**
ATTRIBUTES:			
1	(m)	Attribute:	Address
SERVICES:			
1	(m)	OpsService:	AR-Get Attribute
2	(m)	OpsService:	AR-Set Attribute

6.3.1.2.2 Attributes

Address
This attribute specifies the Data Link Address of Master device.

6.3.1.3 Slave class AREP

6.3.1.3.1 Formal model

ASE:			**AR ASE**
CLASS:			**Slave**
CLASS ID:			**Not used**
PARENT CLASS:			**Top**
ATTRIBUTES:			
1	(m)	Attribute:	PollAddress
2	(m)	Attribute:	UniqueAddress
3	(m)	Attribute:	PreambleLength
SERVICES:			
1	(m)	OpsService:	AR-Get Attribute

2 (m) OpsService: AR-Set Attribute

6.3.1.3.2 Attributes

PollAddress
This attribute specifies the Data Link Address of Slave device. The device responds only to this address in an Identify frame received from a Master device.

Unique address
This attribute specifies the Data Link Address of Slave device. The device responds only to this address in all frames received from a Master device. An all zero value of this attribute is reserved as Broadcast address.

PreambleLength
This attribute specifies the number of preamble octets that have to be transmitted in the beginning of every request PhPDU from Master to this Slave.

6.3.1.4 Application relationship ASE service specifications

6.3.1.4.1 Overview

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

AR Get Attribute
AR Set Attribute

6.3.1.4.2 AR-Get attribute service

6.3.1.4.2.1 Overview

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

6.3.1.4.2.2 Service parameters

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

Table 11 – AR get attributes service parameters

Parameter name	Req	Cnf
Argument	M	
Attribute index	M	
Result (+)		S
Value		C
Result (-)		S
Status		C
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter.		

Argument
The argument carries the parameters of the service request.

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, Unknown value.

6.3.1.4.3 AR Set attributes service

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

6.3.1.4.3.1 Service parameters

The service parameters for the AR Set Attribute Service are shown in Table 12.

Table 12 – AR set attributes service parameters

Parameter name	Req	Cnf
Argument	M	
Attribute index	M	
Value	M	
Result		
Status		M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter.		

Argument
The argument carries the parameters of the service request.

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.

6.4 Summary of classes

This subclause contains a summary of the defined Classes. Table 13 provides a summary of the classes.

Table 13 – Class summary

ASE	Class
Virtual field device	VFD
Data type	Fixed Length Data type
	String Data type
	Structure
	Array
Variable	Simple Variable
	Structure Variable
	Array

6.5 Permitted services by AREP role

Table 14 and Table 15 define the valid combinations of services and AREP class (which service APDUs and AREP with the specified class can send or receive).

Table 14 – Confirmed services by AREP class

Services	Master		Slave	
	req	cnf	Ind	rsp
VFD ASE				
Identify	X	X	X	X
Variable ASE				
Read service – all	X	X	X	X
Write service – all	X	X	X	X
Action ASE	X	X	X	X

Table 15 – Unconfirmed services by AREP class

Services	Slave	Master
	req	ind
Variable ASE		
Information report	X	X

Bibliography

IEC 61158-6-20, *Industrial communication networks – Fieldbus specifications – Part 6-20: Application layer protocol specification – Type 20 elements*

IEC 61784-1, *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*

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

This page deliberately left blank