IEC 61158-5-9

Edition 2.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications –
Part 5-9: Application layer service definition – Type 9 elements

Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 5-9: Définition des services de la couche application – Eléments de type 9

IEC 61158-5-9:2014-08(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

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

IEC 61158-5-9

Edition 2.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 5-9: Application layer service definition – Type 9 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-9: Définition des services de la couche application – Eléments de type 9

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XF

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1735-1

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-9: Application layer service definition – Type 9 elements

FOREWORD

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

Attention is drawn to the fact that the use of the associated protocol type is restricted by its intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a layer protocol type to be used with other layer protocols of the same type, or in other type combinations explicitly authorized by its intellectual-property-right holders.

NOTE Combinations of protocol types are specified in IEC 61784-1 and IEC 61784-2.

International Standard IEC 61158-5-9 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2007. This edition constitutes a technical revision. The main change with respect to the previous edition is listed below:

- Correct download state machine event service

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

FDIS	Report on voting
65C/763/FDIS	65C/773/RVD

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

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

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

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

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

INTRODUCTION

This part of IEC 61158 is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the “three-layer” fieldbus reference model described in IEC 61158-1.

The application service is provided by the application protocol making use of the services available from the data-link or other immediately lower layer. This standard defines the application service characteristics that fieldbus applications and/or system management may exploit.

Throughout the set of fieldbus standards, the term “service” refers to the abstract capability provided by one layer of the OSI Basic Reference Model to the layer immediately above. Thus, the application layer service defined in this standard is a conceptual architectural service, independent of administrative and implementation divisions.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-9: Application layer service definition – Type 9 elements

1 Scope

1.1 General

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

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 9 fieldbus. The term "time-critical" is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible service provided by the different Types of the fieldbus Application Layer in terms of

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

b) the primitive actions and events of the service;

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

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

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

1) the FAL user at the boundary between the user and the Application Layer of the Fieldbus Reference Model, and

2) Systems Management at the boundary between the Application Layer and Systems Management of the Fieldbus Reference Model.

This standard specifies the structure and services of the 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).

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

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the applications are not specified; only a definition of what requests and responses they can

send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this standard to provide access to the FAL to control certain aspects of its operation.

1.2 Specifications

The principal objective of this standard is to specify the characteristics of conceptual application layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of application layer protocols for time-critical communications.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of services standardized as the various types of IEC 61158.

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

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

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

1.3 Conformance

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

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill the Type 9 application layer services as defined in this standard.

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*

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

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

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

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

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

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

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

3 Terms, definitions, symbols, abbreviations and conventions

For the purposes of this document, the following terms, definitions, symbols, abbreviations and conventions as defined in these publications apply.

3.1 ISO/IEC 7498-1 terms

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

3.2 ISO/IEC 8822 terms

a) abstract syntax
b) presentation context

3.3 ISO/IEC 9545 terms

a) application-association
b) application-context
c) application context name
d) application-entity-invocation
e) application-entity-type
f) application-process-invocation
g) application-process-type
h) application-service-element
i) application control service element

3.4 ISO/IEC 8824-1 terms

a) object identifier
b) type

3.5 IEC 61158-1 terms

For the purposes of the present document, the following IEC 61158-1 terms apply.

3.5.1
application
function or data structure for which data is consumed or produced

3.5.2
application layer interoperability
capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.5.3
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.

3.5.4
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.5.5
application process identifier
component that distinguishes multiple application processes used in a device

3.5.6
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.

3.5.7
application process object class
class of application process objects defined in terms of the set of their network-accessible attributes and services

3.5.8
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 preconfiguration activities.

3.5.9
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.5.10
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.

3.5.11
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.

3.5.12
behaviour
indication of how the object responds to particular events

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

3.5.13
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.

3.5.14
class attributes
attribute that is shared by all objects within the same class

3.5.15
class code
unique identifier assigned to each object class

3.5.16
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.

3.5.17
client
(a) object which uses the services of another (server) object to perform a task

(b) initiator of a message to which a server reacts, 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

3.5.18
conveyance path
unidirectional flow of APDUs across an application relationship

3.5.19
cyclic
events which repeat in a regular and repetitive manner

3.5.20
dedicated AR
AR used directly by the FAL User

Note 1 to entry: On Dedicated ARs, only the FAL Header and the user data are transferred.

3.5.21
device
physical hardware connection to the link

Note 1 to entry: A device may contain more than one node.

3.5.22
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.5.23
dynamic AR
AR that requires the use of the AR establishment procedures to place it into an established state

3.5.24
endpoint
one of the communicating entities involved in a connection

3.5.25
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.5.26
error class
general grouping for error definitions

Note 1 to entry: Error codes for specific errors are defined within an error class.

3.5.27
error code
identification of a specific type of error within an error class

3.5.28
FAL subnet
networks composed of one or more data link segments

Note 1 to entry: They are permitted to contain bridges, but not routers. FAL subnets are identified by a subset of the network address.

3.5.29
logical device
FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.5.30
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.

3.5.31
network
series of nodes connected by some type of communication medium

Note 1 to entry: The connection paths between any pair of nodes can include repeaters, routers and gateways.

3.5.32
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.5.33
pre-defined AR endpoint
AR endpoint that is defined locally within a device without use of the create service

Note 1 to entry: Pre-defined ARs that are not pre-established are established before being used.

3.5.34
pre-established AR endpoint
AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.5.35
publisher
role of an AR endpoint in which it transmits APDUs onto the fieldbus for consumption by one or more subscribers

Note 1 to entry: The publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated AR. Two types of publishers are defined by this standard, Pull Publishers and Push Publishers, each of which is defined separately.

3.5.36
server
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

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

3.5.37
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.

3.5.38
subscriber
role of an AREP in which it receives APDUs produced by a publisher

Note 1 to entry: Two types of subscribers are defined by this standard, pull subscribers and push subscribers, each of which is defined separately.

3.6 Type 9 fieldbus application-layer specific definitions

There are no additional terms defined.

3.7 Abbreviations and symbols

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship

AREP	Application Relationship End Point
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Cnf	Confirmation
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLCEP	Data Link Connection End Point
DLL	Data Link Layer
DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	DL-service-data-unit
FAL	Fieldbus Application Layer
ID	Identifier
IEC	International Electrotechnical Commission
Ind	Indication
LME	Layer Management Entity
OSI	Open Systems Interconnect
QoS	Quality of Service
Req	Request
Rsp	Response
SAP	Service Access Point
SDU	Service Data Unit
SMIB	System Management Information Base
SMK	System Management Kernel
VFD	Virtual Field Device

3.8 Conventions

3.8.1 Overview

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

The class specification defines the attributes of the class. The attributes are accessible from instances of the class using the Object Dictionary ASE services specified in 6.3.2 of this standard. The service specification defines the services that are provided by the ASE.

3.8.2 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. Class IDs between 1 and 255 are reserved by this standard to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2 048 are allocated for identifying user defined classes.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.8.3 Conventions for service definitions

3.8.3.1 General

This standard 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.

3.8.3.2 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.

NOTE 1 See the note under 3.8.3.3 relative to the non-inclusion of service parameters that are appropriate to a protocol specification or programming interface specification or implementation specification, but not to an abstract service definition.

This standard 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 standard. 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 2 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 3 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.

3.8.3.3 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 of standards 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 standard but not in an IEC 61158-5 subseries abstract service definition standard. 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 standard or for a protocol standard, except at a level of abstraction that is necessarily common to all embodiments of the specifying standard. 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 standard are detailed in IEC 61158-1, Clause 9.

5 Data type ASE

5.1 Overview

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 Clause 5 are provided in IEC 61158-6-9.

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

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

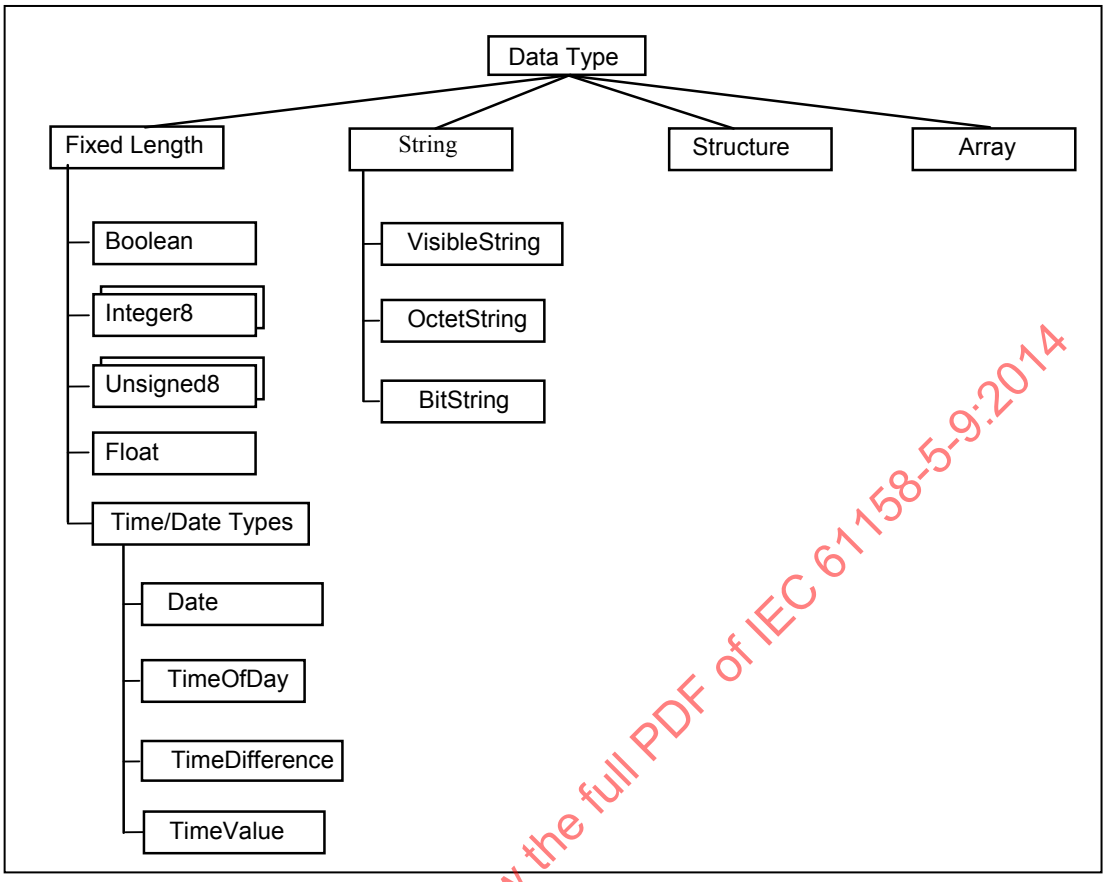


Figure 1 – Data type class hierarchy

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 used to define fixed length and bitstring 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 standard are strings, arrays and structures. There are no standard types defined for arrays and structures.

5.1.1 Overview of basic types

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

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

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

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

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

5.1.1.1 Fixed length types

The length of Fixed length types is an integral number of octets.

5.1.2 Overview of constructed types

Constructed data types are needed to completely convey the variety of information present on the Fieldbus. There are two kinds of constructed types defined for this standard, arrays and structures.

5.1.2.1 Strings

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

5.1.2.2 Arrays

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

5.1.2.3 Structures

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

5.1.2.4 Nesting level

This standard permits arrays and structures to contain arrays and structures. It places a restriction on the number of nesting levels allowed to one.

5.1.3 Specification of user defined data types

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

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

5.1.4 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 61158-6-9. 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:			5 (FIXED LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)
PARENT CLASS:			TOP
ATTRIBUTES:			
1	(m)	Key Attribute:	Data type Numeric Identifier
2	(o)	Key Attribute:	Data type Name
3	(m)	Attribute:	Format (FIXED LENGTH, STRING, STRUCTURE, ARRAY)
4	(c)	Constraint:	Format = FIXED LENGTH | STRING
4.1	(m)	Attribute:	Octet Length
5	(c)	Constraint:	Format = STRUCTURE
5.1	(m)	Attribute:	Number of Fields
5.2	(m)	Attribute:	List of Fields
5.2.1	(o)	Attribute:	Field Name
5.2.2	(m)	Attribute:	Field Data type
6	(c)	Constraint:	Format = ARRAY
6.1	(m)	Attribute:	Number of Array Elements
6.2	(m)	Attribute :	Array Element Data type

5.2.1.1 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 supported by identifying the field by number. This attribute is present when the value of the format attribute is "STRUCTURE".

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 either by referencing a constructed data type definition by its numeric id, or by

embedding a constructed data type definition here. When embedding a description, the Embedded Data type description shown below is used.

Number of Array Elements

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 either by referencing a constructed data type definition by its numeric id, or by embedding a constructed data type definition here. When embedding a description, the Embedded Data type description shown below is used.

Embedded Data type Description

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

1	(m)	Attribute:	Format(FIXED LENGTH, STRING, STRUCTURE, ARRAY)
2	(c)	Constraint:	Format = FIXED LENGTH | STRING
2.1	(m)	Attribute:	Data type Numeric ID value
2.2	(m)	Attribute:	Octet Length
3	(c)	Constraint:	Format = STRUCTURE
3.1	(m)	Attribute:	Number of Fields
3.2	(m)	Attribute:	List of Fields
3.2.1	(m)	Attribute:	Embedded Data type Description
4	(c)	Constraint:	Format = ARRAY
4.1	(m)	Attribute:	Number of Array Elements
4.2	(m)	Attribute:	Embedded Data type Description

5.3 FAL defined data types

5.3.1 Fixed length types

5.3.1.1 Boolean

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

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

5.3.1.2 Integer8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2
2	Data type Name	=	Integer8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

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

5.3.1.3 Integer16

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.4 Integer32

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.5 Unsigned8

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.6 Unsigned16

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.7 Unsigned32

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.8 Float

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

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

5.3.1.9 Date

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	11
2	Data type Name	=	Date
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

This data type is consists of years, months, days or month, days of week, hours, standard/daylight savings time, minutes, and milliseconds.

5.3.1.10 TimeOfDay

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date.

5.3.1.11 TimeDifference

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

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

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	21
2	Data type Name	=	Time Value
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This simple type expresses the time or time difference in a two's complement binary number with a length of eight octets. The unit of time is 1/32 millisecond.

5.3.1.12 TimeValue

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	21
2	Data type Name	=	Time Value
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This simple type expresses the time or time difference in a two's complement binary number with a length of eight octets. The unit of time is 1/32 millisecond.

5.3.2 String types

5.3.2.1 VisibleString

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

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

5.3.2.2 OctetString

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

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet. IEC 61158-6-9 defines the order of transmission.

5.3.2.3 BitString

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

This string type is defined as a series of eight bits, numbered from 0 to 7.

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. The Class ID values have been assigned to be compatible with existing standards. Table 1 is sorted by class id.

Table 1 – Data type summary

Class Id	Data type
1	Boolean
2	Integer8
3	Integer16
4	Integer32
5	Unsigned8
6	Unsigned16
7	Unsigned32

Class Id	Data type
8	Float
9	VisibleString
10	OctetString
11	Date
12	TimeOfDay
14	BitString
21	TimeValue

6 Communication model specification

6.1 Concepts

The concepts are those of the basic communications model described in IEC 61158-1.

6.2 Common parameters

Many services have the following parameters. Instead of defining them with each service, the following common definitions are provided.

AREP ID

This parameter specifies sufficient information to locally identify the AR used to convey the service.

VCR

This parameter specifies sufficient information to locally identify the VCR and AR used to convey the service.

Error Type

This parameter specifies information about the reason for the unsuccessful execution of the service.

6.3 ASEs

6.3.1 Virtual field device ASE

6.3.1.1 Virtual field device class specification

6.3.1.1.1 Class overview

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. The collection of a VFDs object descriptions is defined as the Object Dictionary (OD). For each VFD there is exactly one Object Dictionary. Services are defined for accessing a VFD's OD and its APOs, as shown in Figure 2.

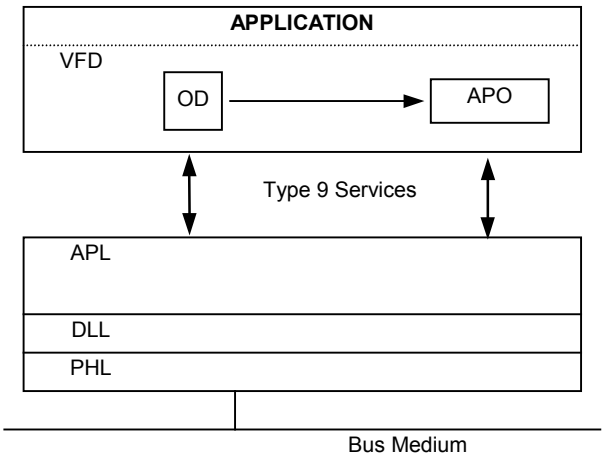


Figure 2 – VFD model

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 standard. It should only be indicated, how the abstractly described services may be made available to the application.

Access to VFD objects is defined by Virtual Communication Relationships (VCR). Several VFD Objects may be present in a device, as shown in Figure 3.

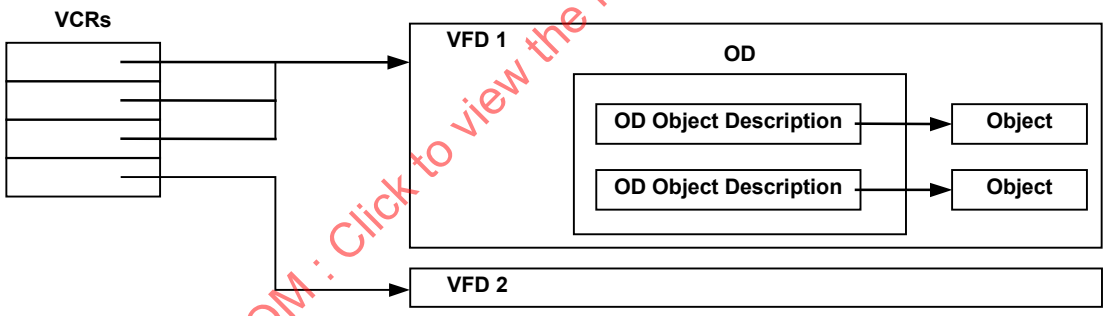


Figure 3 – Abstract model of an automation system (VFD)

6.3.1.1.2 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.	(m)	Key Attribute:	implicit through the VCR
2.	(m)	Attribute:	VendorName
3.	(m)	Attribute:	ModelName
4.	(m)	Attribute:	Revision
5.	(m)	Attribute:	ProfileNumber
6.	(m)	Attribute:	LogicalStatus
7.	(m)	Attribute:	PhysicalStatus
8.	(m)	Attribute:	List of VFD Specific Objects

SERVICES:

1. (m) Ops Service: Status
2. (m) Ops Service: Unsolicited Status
3. (m) Ops Service: Identify

6.3.1.1.3 Attributes

Implicit through the VCR

The VCR which is entered in the VCR List points to the assigned VFD Object.

VendorName

This attribute identifies the vendor of the device.

ModelName

This attribute specifies the model name of the device. The value of this attribute is defined by the vendor.

Revision

This attribute describes the revision level of the device. The value of this attribute is defined by the vendor.

ProfileNumber

This attribute identifies the profile of the VFD. The profile value is defined by an external administrative authority. If the device does not correspond to a profile, then this attribute has a null value.

Logical Status

This attribute specifies information about the state of the communication capabilities of the device. Its values are shown in Table 2.

Table 2 – Logical status

Logical status	Meaning
Ready for communication	All services may be used normally.
Limited number of services	Limited number of services may be supported for some cases.
OD-LOADING-NON-INTERACTING	If the Object Dictionary is in the state OD-LOADING-NON-INTERACTING, it is not allowed to execute the service InitiatePutOD.
OD-LOADING-INTERACTING	If the Object Dictionary is in the state OD-LOADING-INTERACTING, then all connections, except for the connection over which the InitiatePutOD Service was received, are locked and the establishment of a further connection should be rejected. On this connection only the following services are allowed: Initiate PhysWrite Abort GetOD Reject InitiatePutOD Status PutOD Identify TerminatePutOD PhysRead

Physical Status

This attribute gives a coarse summary of the state of the real device. Its values are:

- operational
- partially operational
- not operational
- needs maintenance

List of VFD Specific Objects

Contains all VFD Specific Objects.

6.3.1.1.4 Services

Status

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

Unsolicited Status

This service is used by the AP to report its network visible state.

Identify

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

6.3.1.2 Virtual field device ASE service specification

6.3.1.2.1 Supported services

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

- Status
- Unsolicited Status
- Identify

6.3.1.2.2 Status service

6.3.1.2.2.1 Service overview

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

6.3.1.2.2.2 Service primitives

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

Table 3 – Status

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Result(+)			S	S (=)
Logical Status			M	M (=)
Physical Status			M	M (=)
Local Detail			U	U (=)
Result(-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Logical Status

This parameter specifies the value of the Logical Status attribute of the VFD.

Physical Status

This parameter specifies the value of the Physical Status attribute of the VFD.

Local Detail

This parameter specifies the user defined state of the application.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.1.2.2.3 Service procedure

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

6.3.1.2.3 Unsolicited status service

6.3.1.2.3.1 Service overview

The Status Notification service is used by the AP to report its state spontaneously.

6.3.1.2.3.2 Service primitives

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

Table 4 – Unsolicited status

Parameter name	Req	Ind
Argument		
VCR	M	M
Destination DL-Address	C	
Source DL-Address		C
LogicalStatus	M	M (=)
Physical Status	M	M (=)
Local Detail	U	U (=)

Argument

The argument contains the parameters of the service request.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and the Remote Address Configuration Type is FREE. It gives the remote address to which the Status Notification should be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and the Remote Address Configuration Type is FREE. This parameter indicates the remote address from which the indicated Status Notification is to be sent.

Logical Status

This parameter specifies the value of the Logical Status attribute of the VFD.

Physical Status

This parameter specifies the value of the Physical Status attribute of the VFD.

Local Detail

This parameter specifies the user defined state of the application.

6.3.1.2.3.3 Service procedure

The Unconfirmed Service Procedure specified in Clause 4 applies to this service.

6.3.1.2.4 Identify service

6.3.1.2.4.1 Service overview

Information to identify an AP is read with this service.

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

6.3.1.2.4.2 Service primitives

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

Table 5 – Identify

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Result(+)			S	S (=)
Vendor Name			M	M (=)
Model Identifier			M	M (=)
Vendor Revision			M	M (=)
Result(-)			S	S (=)
Error Type			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. See 1.2.				

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Vendor Name

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

Model Identifier

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

Vendor Revision

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

Result(-)

This selection type parameter indicates that the service request failed.

6.3.1.2.4.3 Service procedure

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

6.3.2 Object dictionary (OD) ASE

6.3.2.1 Overview

The ASE Models each specify one or more related APO classes as a collection of attributes and services.

The collection of attributes for the APOs of an AP is referred to as the Object Dictionary (OD). The attributes for a single APO is referred to as its object description. To access the object descriptions of APOs, this ASE defines common OD services. These services operate using the client/server model.

The Object Dictionary (OD) contains the Object Descriptions of the following Communication Objects:

- DataType
- Domain
- SimpleVariable
- Record
- Event
- DataTypeStructureDescription
- ProgramInvocation
- Array
- VariableList

An object description "Object Dictionary" (OD object description) is uniquely assigned to the Object Dictionary. The OD object description contains information about the structure of the Object Dictionary.

The object descriptions are marked with an unique index of type Unsigned16.

A name of type visible string may also be assigned to the following objects:

- Domain
- SimpleVariable
- Record
- Event
- ProgramInvocation
- Array
- VariableList

The index or the name serves as a key to the object and the object description.

The services address on principle the object of the server, because the real object exists there.

Before a remote communication partner can access an object, it needs to know its description. The logical address (Index) or the name may be used, when objects are accessed remotely.

The object descriptions may be specified during system configuration, but they may also be transferred between stations at any later time.

The definition of the object description takes place at that station, in which the object really exists (Source OD). The length of an object description may be different for the individual objects. The communication partners maintain a complete or a partial copy of the Remote Object Description (Remote OD). This is illustrated in Figure 4.

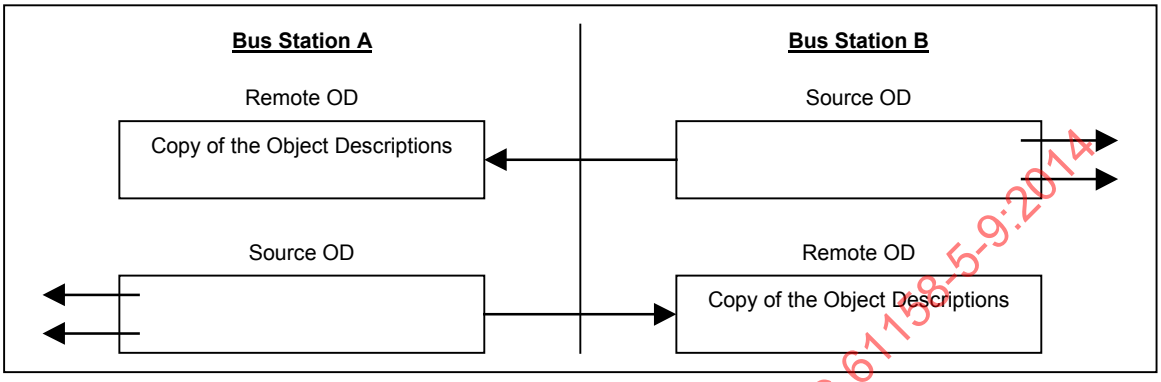


Figure 4 – Source OD/remote OD

Thus every station may possess a Source OD for locally existing communication objects and one or more Remote ODs. The Remote ODs are connection specific, i.e. for each connection another Remote OD may be valid.

The OD model provides corresponding services for reading and writing of Source ODs. If a Source OD is modified by a remote communication partner, then this should be notified to all other communication partners which use the same data base (Source OD). For this purpose, the user releases the connections to all other communication partners. After the OD has been modified, the connections may be established again. During the connection establishment, the new version number (Version OD) is given to each communication partner to notify that the corresponding Source OD has been modified. The addition of object descriptions is permissible without connection release.

The structure of the OD described as follows does not prescribe a concrete implementation of the OD. It only defines the structure of the object descriptions which are transmitted via the bus. Even the existence of an OD is not necessary, if the needed information can be obtained in another way (e.g. by an algorithm). Detailed attributes and transfer syntaxes are specified in the appropriate models.

The Object Dictionary (OD) is divided into several parts, as shown in Table 6:

– The Object description of the "Object Dictionary" (OD object description) contains structural information about the OD.
– Static Type Dictionary (ST-OD) contains Data types and Data type Structure Description.
– Static Object Dictionary (S-OD) contains object descriptions of Simple Variables, Arrays, Records, Domains and Events.
– Dynamic List of Variable Lists (DV-OD) contains object descriptions of variable lists.
– Dynamic List of Program Invocations (DP-OD) contains object descriptions of Program Invocations.

Table 6 – Structure of the object dictionary

Index	Object Dictionary (OD)
0	OD object description (OD-ODES)
1 ... i	Static List of Types (ST-OD)
k ... n	Static Object Dictionary (S-OD)
p ... t	Dynamic List of Variable Lists (DV-OD)
v ... z	Dynamic List of Program Invocations (DP-OD)

6.3.2.1.1 Static list of types (ST-OD)

The Static List of Types (ST-OD) contains the object descriptions of Data types and Data type Structure Descriptions for the Variable Access objects of the S-OD, as shown in Table 7.

Table 7 – Structure of the static list of types

Index	Static List of Types (ST-OD)
1 ... c	contains objects: Data types
d ... i	contains objects: Data types and Data type Structure Descriptions

Exactly one object description is assigned to every index. This assignment is static and may not be deleted. The ST-OD may be loaded by the remote partner.

The Static List of Types always begins with the index 1 in the Object Dictionary. The remaining object descriptions should follow with increasing index. The number of assignments "index <=> object description" is entered as length of the ST-OD in the Object Dictionary.

The ST-OD is divided in two segments. The first segment contains the object descriptions of standardized Data types (Standard Data types) and begins with index 1. The second segment follows immediately and contains object descriptions of Data types and Data type Structure Descriptions, which can be defined individually. This segment may be further specified by user groups with the help of profiles.

6.3.2.1.2 Static object dictionary (S-OD)

The Static Object Dictionary (S-OD) contains the object descriptions of Simple Variables, Arrays, Records, Domains and Events, as shown in Table 8.

Table 8 – Structure of the static object dictionary

Index	Static Object Dictionary (S-OD)
k ... n	contains objects: Simple Variables, Arrays, Records, Domains and Events

An object description is assigned to each index. The first index, which is associated to an object description, is entered in the object description of the Object Dictionary. The remaining object descriptions should follow with increasing index. The number of assignments "index <=> object description" is entered as length of the S-OD in the object description of Object Dictionary.

Besides the assignment "index <=> object description", an additional assignment "name <=> object description" may exist. The length of the name may be from 0 to 32 octets. It is entered in the Name Length Field of the OD object description. If this field contains a "0", no names are present. The objects which are defined in the S-OD may be provided with access rights. Whether Access Rights are possible is entered in the field AccessProtectionSupported of the OD object description.

An object description in the S-OD contains an extension, which may contain further object specific definitions.

6.3.2.1.3 Dynamic list of variable lists (DV-OD)

The Dynamic List of Variable Lists (DV-OD) contains the object description of the Variable Lists, as shown in Table 9.

Table 9 – Structure of the dynamic list of variable lists

Index	Dynamic List of Variable Lists (DV-OD)
p ... t	contains objects: Variable List

An object "Variable List" and its object description in the DV-OD is dynamically created with the service DefineVariableList and may be deleted with the service DeleteVariableList. Upon creation of the object access rights may be assigned to it. Moreover it is possible, to create these objects by loading the whole OD.

The first index, which has the object description of a Variable List assigned to it, and the number of available indices are entered in the OD object description.

Fundamental information is entered in the object description about the Variable List, for example access rights, the number of Variable Access objects and the logical addresses (indices) of Variable Access objects in the S-OD.

Besides the assignment "index <=> object description", an additional assignment "name <=> object description" may exist.

6.3.2.1.4 Dynamic list of program invocations (DP-OD)

The Dynamic List of Program Invocations (DP-OD) contains the object descriptions of the objects Program Invocation, as shown in Table 10.

Table 10 – Structure of the dynamic list of program invocations

Index	Dynamic List of Program Invocations (PD-OD)
v ... w	Predefined Program Invocations
x ... z	Dynamic Program Invocations

An object Program Invocation and its object description in the DP-OD is dynamically created with the service Create Program Invocation and deleted with the service Delete Program Invocation. Upon creation of the object access rights may be assigned to it. Moreover it is possible to create these objects by loading the whole OD.

The first index, which has the object description of a Program Invocation assigned to it, and the number of available indices are entered in the OD object description.

The object description of a Program Invocation contains information such as access rights and the number of Domain objects and their logical addresses (indices).

Besides the assignment "index <=> object description", an additional assignment "name <=> object description may exist.

The DP-OD may contain a segment with pre-configured Program Invocations.

6.3.2.2 OD class specifications

6.3.2.2.1 Class overview

Every object description contains an index, an object code, object attributes, system specific references to the real object and, if need be, a name and an extension.

The object is addressed by the index and identified by the object description. The object code is the identifier of the object and indicates which class this object belongs to. The other object attributes are object specific. The content of the extension field is application-specific.

The local address serves the internal addressing of the object. Besides the index, optionally a name for the addressing of the object may also be used. The extension contains a length information for the extension and optionally further object attributes. The value "0" for the length means: no further object attributes.

6.3.2.2.2 OD description class specification

6.3.2.2.2.1 Formal model

The structure of the Object Dictionary is described by the first entry in the OD. This has the index "0" in the Object Dictionary. By reading the OD object description the structure description and the version number of the OD are made available.

ASE:	**Object Dictionary ASE**
CLASS:	**OD Description**
CLASS ID:	**—**
PARENT CLASS:	**TOP**
ATTRIBUTES:	
1. (m) Key Attribute:	OD Index
2. (m) Attribute:	ROM/RAM Flag
3. (m) Attribute:	Name Length
4. (m) Attribute:	Access Protection Supported
5. (m) Attribute:	Version OD
6. (m) Attribute:	Local Address OD-ODES
7. (m) Attribute:	ST-OD Length
8. (m) Attribute:	LocalAddress ST-OD
9. (m) Attribute:	First Index S-OD
10. (m) Attribute:	S-OD Length
11. (m) Attribute:	Local Address S-OD
12. (m) Attribute:	First Index DV-OD

13. (m) Attribute: DV-OD Length
14. (m) Attribute: LocalAddress DV-OD
15. (m) Attribute: First Index DP-OD
16. (m) Attribute: DP-OD Length
17. (m) Attribute: Local Address DP-OD

SERVICES:

1. (m) OpsService: Get OD
2. (m) OpsService: Initiate Put OD
3. (m) OpsService: Put OD
4. (m) OpsService: Terminate Put OD

6.3.2.2.2.2 Attributes

OD Index

OD Index of the OD Object Description, here always = 0.

ROM/RAM Flag

This attribute of type Boolean describes whether modifications in the OD are allowed.

false <=> no modifications in the OD are permitted

true <=> modifications in the OD are permitted

Name Length

This field gives the length of the name and may take only the values "0..32":

0 <=> no names are used

Access Protection Supported

This attribute states, whether access rights for pass-word, access groups and all communication partners are supported (true), or whether the access to all objects is permitted for every communication partner (false).

Version OD

Gives the version of the OD.

Local Address OD-ODES

This attribute is a system specific reference to the real OD object description and serves the internal addressing. If no representation of this kind is used, this attribute is set to the value FFFFFFFF hex.

ST-OD Length

This attribute gives the maximal number of available entries in the Static List of Types.

NOTE The first available index of the Static List of Types is always = 1, therefore the highest available index in the Static List of Types is equal to its length.

Local Address ST-OD

This attribute is a system specific reference to the real ST-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

First Index S-OD

The first available index in the Static Object Dictionary.

S-OD Length

This attribute gives the maximum number of available entries in the Static Object Dictionary.

NOTE The highest available index is S-OD-Length = FirstIndex-S-OD + S-OD-Length - 1 (done)

Local Address S-OD

This attribute is a system specific reference to the real S-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

First Index DV-OD

First available index in the Dynamic List of Variable Lists.

0 <=> no Dynamic List of Variable Lists.

DV-OD Length

This attribute gives the maximum number of available entries in the Dynamic List of Variable Lists.

NOTE The highest available index is DV-OD-Length = FirstIndex-DV-OD + OD-OD-Length – 1.

Local Address DV-OD

This attribute is a system specific reference to the real DV-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

First Index DP-OD

First available index in the Dynamic List of Program Invocations.

0 <=> no Dynamic List of Program Invocations.

DP-OD Length

This attribute gives the maximum number of available entries in the Dynamic List of Program Invocations.

NOTE The highest available index is DP-OD-Length = FirstIndex-DP-OD + DP-OD-Length – 1.

Local Address DP-OD

This attribute is a system specific reference to the real DP-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

6.3.2.2.2.3 Services

All services defined for this class are mandatory.

Get OD

This confirmed service is used to read entries of the OD.

Initiate Put OD

This confirmed service is used to begin the process of writing entries of the OD.

Put OD

This confirmed service is used to write entries of the OD.

Terminate Put OD

This confirmed service is used to terminate the process of writing entries of the OD.

6.3.2.2.2.4 Empty object dictionary

The Object Dictionary is loadable. Therefore if an Object Dictionary is not yet loaded, at least an empty Object Dictionary should exist.

An empty Object Dictionary is defined by pre-setting the attributes of the OD object description shown in Table 11:

Table 11 – Empty object dictionary

Attribute	Value
ROM/RAM-Flag	True
NameLength	0
AccessProtectionSupported	False
VersionOD	0
ST-OD-Length	0
FirstIndex-S-OD	0
S-OD-Length	0
FirstIndex-DV-OD	0
DV-OD-Length	0
FirstIndex-DP-OD	0
DP-OD-Length	0

6.3.2.2.3 OD object formal model

6.3.2.2.3.1 Formal model

The structure of the Object Dictionary is described by the first entry in the OD. This has the index "0" in the Object Dictionary. By reading the OD object description the structure description and the version number of the OD are made available.

ASE: **Object Dictionary ASE**
CLASS: **OD Object**
CLASS ID: **—**
PARENT CLASS: **TOP**
ATTRIBUTES:
1. (m) Key Attribute: implicit by VFD
2. (m) Attribute: OD object description
3. (m) Attribute: List of object description
4. SERVICES:

1. (m) OpsService: Get OD
2. (m) OpsService: Initiate Put OD
3. (m) OpsService: PutOD
4. (m) OpsService: Terminate Put OD

6.3.2.2.3.2 Attributes

implicit by VFD

The VCR, which is entered in the VCR List points to the assigned Object Dictionary.

OD object description

Contains the object description of the Object Dictionary.

List of object description

Contains object descriptions of the following objects:

Domain

ProgramInvocation

SimpleVariable

Array

Record

VariableList

DataType

DataTypeStructureDescription

Event

6.3.2.2.3.3 Services

All services defined for this class are mandatory.

Get OD

This confirmed service is used to read entries of the OD.

Initiate Put OD

This confirmed service is used to begin the process of writing entries of the OD.

Put OD

This confirmed service is used to write entries of the OD.

Terminate Put OD

This confirmed service is used to terminate the process of writing entries of the OD.

6.3.2.3 Object dictionary model services

6.3.2.3.1 Supported services

OD services are specified to read or write the object descriptions of Domain, Program Invocation, Simple Variable, Array, Record, Variable List, Event, Object Dictionary, Data type and Data type Structure Description objects. The attributes of an APO which can be written are specified as part of the attribute definition.

Because unexpected changes to the object definitions within an AP can affect operation of the AP, the Initiate Put OD and the Terminate Put OD services are defined. The Initiate Put OD service is used to request the remote AP for permission to change the OD. The Terminate Put OD service is used to indicate to the remote AP that changes to its OD are complete and that it can resume normal operation.

With the GetOD service one or several object descriptions are read. With the PutOD service one or several object descriptions are written. The number of object descriptions, which are transferable with one PutOD or GetOD Service, depends on their length and the maximum available PDU size.

With the PutOD services one or several object descriptions are written. Before the beginning of a modification of the OD an user may take appropriate measures (e.g. to put a process into a safe state). The communication does not test the plausibility of the new Object Dictionary.

Writing in the OD is performed with the sequence InitiatePutOD service, one or several PutOD services and a TerminatePutOD service. For this it is distinguished between a loading which is free of interactions and a loading which is not free of interactions (OD-LOADING-NON-INTERACTING and OD-LOADING-INTERACTING). The loading which is not free of interactions differentiates between a new loading (complete) and a post loading (partial). If need be, new communication objects and their state machines are created with the TerminatePutOD.

Loading free of interactions (OD-LOADING-NON-INTERACTING):

The client communicates to the server with the parameter Consequence = 0, whether the intended modifications or extensions are free of interactions for the other communication partners. In this case the VCRs are continued. For example, modifications free of interactions are

– addition of new object descriptions in the static segment,
– deletion of private object descriptions for stations which are no longer in the communication system.

The communication does not check whether the write to the OD is really free of interactions.

The loading free of interaction is initiated with the service InitiatePutOD with the parameter Consequence = 0. After that object descriptions may be loaded with PutOD. As long as the loading process is not terminated with TerminatePutOD, every further InitiatePutOD is rejected.

Loading not free of interactions (OD-LOADING-INTERACTING):

The loading not free of interactions is initiated with the service InitiatePutOD with the Parameter Consequence = 1 (post-loading) or Consequence = 2 (new-loading). In this case the user may abort all active VCRs, except the one over which the InitiatePutOD is received to this VFD. After that the object description may be loaded with PutOD. It is also the task of the client user to enter the new version number in the OD. The loading process is terminated with the service TerminatePutOD. After that all VCRs may be re-established. With the re-establishment of the VCRs the new version number of the OD is transferred. The communication partners may then recognize that the corresponding OD has been modified. Whereas for post-loading solely parts of the OD are loaded, the existing OD is deleted and subsequently loaded completely anew upon receipt of an InitiatePutOD with Consequence = 2.

6.3.2.3.2 Get OD service

6.3.2.3.2.1 Service overview

The service distinguishes between a short and a long form. The long form of the GetOD service is optional. To read a single object description, the index or the name should be given.

To read several or all object descriptions, the index of the first required object description (StartIndex) should be given. To read the whole OD, the GetOD service should be used repeatedly.

6.3.2.3.2.2 Service parameters

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

Table 12 – Get OD service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
All Attributes	M	M (=)		
Access Specification	M	M (=)		
Index	S	S (=)		
Variable Name	S	S (=)		
Variable List Name	S	S (=)		
Domain Name	S	S (=)		
PI Name	S	S (=)		
Event Name	S	S (=)		
Start Index	S	S (=)		
Result(+)			S	S (=)
List of Object Description			M	M (=)
More Follows			M	M (=)
Result(-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the parameters of the service request.

All Attributes

This parameter states whether the object description should be transferred in the short form (false) or in the long form (true).

The following object attributes are not contained in the short form:

- Password
- AccessGroups
- AccessRights
- LocalAddress
- Name
- LocalAddress-OD-ODES
- LocalAddress-ST-OD
- LocalAddress-S-OD
- LocalAddress-DV-OD
- LocalAddress-DP-OD
- Extension

Access Specification

This parameter states which object description is accessed.

Index

Index of the object description.

Variable Name

Name of a variable.

Variable List Name

Name of a Variable List.

PI Name

Name of a Program Invocation.

Domain Name

Name of a Domain.

Event Name

Name of an Event.

StartIndex

Index, starting with which the object descriptions should be read. When this option is selected, the responding VFD sends object descriptions of indexes greater than StartIndex in the order of increasing Index. When all of requested object descriptions are not sent in the response, the MoreFollows flag should be set.

Result(+)

This selection type parameter indicates that the service request succeeded.

List of object description

For logical access and for access by name one, otherwise several complete object descriptions. The number depends on the maximum possible PDU size and the length of the single object descriptions. If access rights are not supported (Access Protection = false), the attributes Password, Access Groups and Access Rights are meaningless.

More Follows

States whether further object descriptions exist after reading several object descriptions selecting StartIndex for Access Specification. This parameter is set true when the responding VFD has at least one object description of Index greater than that of the last object description in the response. It is set false when the responding VFD has no more object descriptions. For the access to single object descriptions selecting other options for Access Specification, descriptions this parameter is always set to false.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.2.3.2.3 Service procedure

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

If the user is able to access the specified object descriptions, and if they can be conveyed in a single APDU, they are returned in the response (+) primitive.

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

If the user is unable to get the description for the first object in the list, the service fails and the user issues a response (-) primitive indicating the reason.

6.3.2.3.3 InitiatePutOD

6.3.2.3.3.1 Service overview

The InitiatePutOD opens the process of loading the OD free of interaction or not free of interaction.

6.3.2.3.3.2 Service parameters

The service parameters for this service are shown in Table 13.

Table 13 – Initiate put OD service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Consequence	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the parameters of the service request.

Consequence

This parameter states whether the intended modifications are free of interaction for the other communication partners, not free of interaction with partial reload of the OD, or not free of interaction with completely new load of the OD.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.2.3.3.3 Service procedure

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

6.3.2.3.4 Put OD service

6.3.2.3.4.1 Service overview

With the PutOD service one or several object descriptions are written into the OD.

6.3.2.3.4.2 Service parameters

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

Table 14 – Put OD service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
List of Object Description	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the parameters of the service request.

List of Object Description

One or several object descriptions which should be entered into the OD. If access rights are not supported (Access Protection = false), the attributes Password, Access Groups and Access Rights are meaningless.

An object description may also be deleted with the PutOD service. In this case, an empty object description with the following structure is transferred:

– Index of the object description to be deleted
– Object Code = Null Object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.2.3.4.3 Service procedure

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

6.3.2.3.5 Terminate put OD service

6.3.2.3.5.1 Service overview

The Terminate Put OD service terminates the process of loading of the OD. The objects with their state machines are generated.

6.3.2.3.5.2 Service parameters

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

Table 15 – Terminate put OD service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)
Index			C	C (=)

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. See 1.2.

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Index

This parameter gives the Index of the object, for which the generation was not successful.

6.3.2.3.5.3 Service procedure

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

6.3.2.3.6 Put OD services state machine

The state machine is shown in Figure 5.

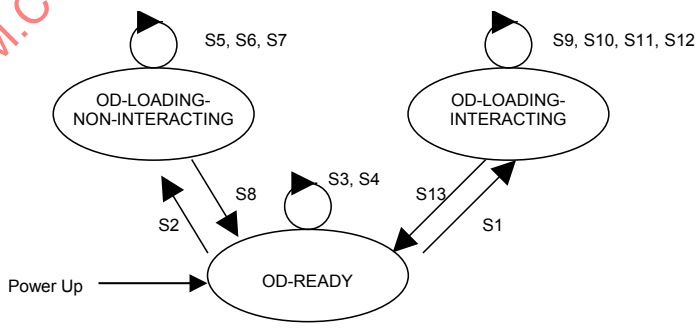


Figure 5 – Put OD state machine

6.3.2.3.6.1 States

OD-LOADING-NON-INTERACTING

In this state the execution of the service InitiatePutOD is not permitted. In this state the service PutOD on the DV-OD or the DP-OD is not permitted.

OD-LOADING-INTERACTING

Except for the connection over which the InitiatePutOD service was received, all connections are locked and the establishment of a further connection should be rejected. On this connection only the following services are allowed:

- Initiate
- Abort
- Reject
- Status
- Identify
- PhysRead
- PhysWrite
- GetOD
- InitiatePutOD
- PutOD
- TerminatePutOD

OD-READY

In this state all connections and services may be used normally.

6.3.2.3.6.2 State transitions

Table 16 specifies the state transitions related to PutOD.

Table 16 – Put OD state transitions

#	Current State	Events	Actions	Next State
S1	OD-READY	InitiatePutOD.ind && consequence <> loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-INTERACTING
S2	OD-READY	InitiatePutOD.ind && consequence = loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-NON-INTERACTING

S3	OD-READY	PutOD.ind	PutOD.rsp(-){
			ErrorType = object-state-conflict
			}
				OD-READY

S4	OD-READY	TerminatePutOD.ind	TerminatePutOD.rsp(-){
			ErrorType = object-state-conflict
			}
				OD-READY

S5	OD-LOADING-NON-INTERACTING	InitiatePutOD.ind	InitiatePutOD.rsp(-){
			ErrorType = object- state-conflict
			}
				OD-LOADING-NON-INTERACTING

S6	OD-LOADING-NON-INTERACTING	PutOD.ind	PutOD.rsp(+){ }	OD-LOADING-NON-INTERACTING

S7	OD-LOADING-NON-INTERACTING	TerminatePutOD.ind
		&& OD is not usable
			TerminatePutOD.rsp(-){
			ErrorType = operational-problem
			}
				OD-LOADING-NON-INTERACTING

S8	OD-LOADING-NON-INTERACTING	TerminatePutOD.ind && OD is okay	TerminatePutOD.rsp(+){ }	OD-READY
S9	OD-LOADING-INTERACTING	InitiatePutOD.ind && consequence <> loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-INTERACTING

S10	OD-LOADING-INTERACTING	InitiatePutOD.ind
		&& consequence = loading-free-of-interaction
			InitiatePutOD.rsp(-){
			ErrorType = object-state-conflict
			}
				OD-LOADING-INTERACTING

S11	OD-LOADING-INTERACTING	PutOD.ind	PutOD.rsp(+){ }	OD-LOADING-INTERACTING

S12	OD-LOADING-INTERACTING	TerminatePutOD.ind
		&& OD is not usable
			TerminatePutOD.rsp(-){
			ErrorType = operational-problem
			}
				OD-LOADING-INTERACTING

S13	OD-LOADING-INTERACTING	TerminatePutOD.ind && OD is okay	TerminatePutOD.rsp(+){ }	OD-READY

6.3.3 Context management ASE

6.3.3.1 Overview

The context management services are used to explicitly establish a VCR, to release a VCR and to reject improper services. The data necessary for this are stored in the OD object description and in the VCR object. Transaction objects are provided for each VCR. The transaction objects are used for the management of confirmed service primitives.

6.3.3.2 Context management class specifications

6.3.3.2.1 VCR list class specification

6.3.3.2.1.1 VCR list formal model

The Virtual Communication Relationship List (VCR List) contains the descriptions of all VCRs of a device. A VCR consists of static part and dynamic part. This subclause gives the definition to FMS specific part of VCR. Refer Network Management Specification for the definition of other part of VCR.

ASE: **Context Management ASE**
CLASS: **VCR List**
CLASS ID: **—**
PARENT CLASS: **TOP**
VCR STATIC ATTRIBUTES:

1.1	(m)	Attribute:	Static VCR ID
1.2	(m)	Attribute:	Max APDU Size Sending
1.3	(m)	Attribute:	Max APDU Size Receiving
1.4	(m)	Attribute:	Features Supported Length
1.5	(m)	Attribute:	Features Supported
1.6	(m)	Attribute:	Max Outstanding Services Sending
1.7	(m)	Attribute:	Max Outstanding Services Receiving
1.8	(m)	Attribute:	VFD ID

VCR DYNAMIC ATTRIBUTES:

2.1	(m)	Attribute:	Dynamic VCR ID
2.2	(m)	Attribute:	Outstanding Services Counter Client
2.3	(m)	Attribute:	Outstanding Services Counter Server
2.4	(m)	Attribute:	VCR State
2.5	(m)	Attribute:	Actual Max Outstanding Services Sending
2.6	(m)	Attribute:	Actual Max Outstanding Services Receiving

SERVICES:

1.	(m)	Ops Service:	Initiate
2.	(m)	Ops Service:	Abort
3.	(m)	Ops Service:	Reject

6.3.3.2.1.2 Attributes

VCR STATIC ATTRIBUTES:

Static VCR ID

This attribute is the index in the Object Dictionary for the static part of this VCR.

Max APDU Sending

This attribute specifies the maximum length of the APDU that may be sent on this VCR.

Max APDU Receiving

This attribute specifies the maximum length of the APDU that may be received on this VCR.

Features Supported Length

This attribute defines the number of octets present in the Features Supported field.

Features Supported

This attribute identifies the services and features supported on this VCR, as shown in Table 17. This subclause specifies bit assignment of fixed part and the method for extension of services and features. The Features Supported is of the length specified by Features Supported Length. It consists of two concatenated bit strings of the same length. The first bit string identifies services and features as a client and the second bit string identifies services and features as a server. For each optional service there are two bits, one in the first bit string and another in the second bit string. For the optional feature there are two bits, one in the first bit string, specifying the optional feature is permissible. The extension of the attribute can be made by using unassigned bits or adding another octet at the end of each bit string. All bits within an octet, which are not specified, have the value 0.

Table 17 – Attribute FMS features supported

Service	Primitive of client functions	bit(n)	Primitive of server functions	bit(n)
GetOD (Long form)	.req, .cnf	0	.ind, .rsp	0
UnsolicitedStatus	.req	1	.ind	1
InitiatePutOD	.req, .cnf	2	.ind, .rsp	2
PutOD	.req, .cnf	"	.ind, .rsp	"
TerminatePutOD	.req, .cnf	"	.ind, .rsp	"
InitiateDownloadSequence	.req, .cnf	3	.ind, .rsp	3
DownloadSegment	.ind, .rsp	"	.req, .cnf	"
TerminateDownloadSequence	.ind, .rsp	"	.req, .cnf	"
InitiateUploadSequence	.req, .cnf	4	.ind, .rsp	4
UploadSegment	.req, .cnf	"	.ind, .rsp	"
TerminateUploadSequence	.req, .cnf	"	.ind, .rsp	"
RequestDomainDownload	.req, .cnf	5	.ind, .rsp	5
RequestDomainUpload	.req, .cnf	6	.ind, .rsp	6
CreateProgramInvocation	.req, .cnf	7	.ind, .rsp	7
DeleteProgramInvocation	.req, .cnf	"	.ind, .rsp	"
Start	.req, .cnf	8	.ind, .rsp	8
Stop	.req, .cnf	"	.ind, .rsp	
Resume	.req, .cnf		.ind, .rsp	
Reset	.req, .cnf		.ind, .rsp	"
Kill	.req, .cnf	9	.ind, .rsp	9
Read	.req, .cnf	10	.ind, .rsp	10
Write	.req, .cnf	11	.ind, .rsp	11
ReadWithType	.req, .cnf	12	.ind, .rsp	12
WriteWithType	.req, .cnf	13	.ind, .rsp	13
PhysRead	.req, .cnf	14	.ind, .rsp	14
PhysWrite	.req, .cnf	15	.ind, .rsp	15
InformationReport	.req	16	.ind	16
InformationReportWithType	.req	17	.ind	17
DefineVariableList	.req, .cnf	18	.ind, .rsp	18
DeleteVariableList	.req, .cnf	"	.ind, .rsp	"
EventNotification	.req	19	.ind	19
EventNotificationWithType	.req	20	.ind	20
AcknowledgeEventNotification	.req, .cnf	21	.ind, .rsp	21
AlterEventConditionMonitoring	.req, .cnf	22	.ind, .rsp	22
Addressing with Name	.req	23	.ind	23
Extensions	**Primitive of client functions**	**bit[n]**	**Primitive of server functions**	**bit[n]**
GenericInitiateDownloadSequence	.req, .cnf	24	.ind, .rsp	24
GenericDownloadSegment	.req, .cnf	"	.ind, .rsp	"
GenericTerminateDownloadSequence	.req, .cnf	"	.ind, .rsp	"
Explanation: [n] : 0 to (23+8x); without extension ==> x = 0; with extension ==> x = 1,2, ...				

Following services are mandatory for all VCRs to open and close them:

– Initiate – Abort

For consistency, the server supports at least the following basic services:

– Identify – Status
– Reject – GetOD

Max Outstanding Services Sending (MaxSCC)

This attribute specifies the maximum number of outstanding confirmed services capable in this station as the requester side of this VCR.

Max Outstanding Services Receiving (MaxRCC)

This attribute specifies the maximum number of outstanding confirmed services capable in this station as the responder side of this VCR.

VFD Reference

This attribute identifies the VFD that uses the VCR.

VCR DYNAMIC ATTRIBUTES:

Outstanding Services Counter Client (OSCS)

This attribute should specify the number of currently outstanding confirmed services on this VCR as a requester.

Outstanding Services Counter Server (OSCR)

This attribute should specify the number of outstanding services that are currently being processed on this VCR as a responder.

VCR State

This attribute should specify the state of the VCR. A connection-oriented VCR may be in one of the following states:

– CONNECTION-NOT-ESTABLISHED
– CONNECTION-ESTABLISHING (CALLING)
– CONNECTION-ESTABLISHING (CALLED)
– CONNECTION-ESTABLISHED

A connectionless VCR may be in one of the following states:

– CONNECTIONLESS-CLIENT
– CONNECTIONLESS-SERVER

Actual Max Outstanding Services Sending (ActualMaxSCC)

This attribute specifies the maximum number of outstanding confirmed services allowed in this station as the requester side of this VCR. Its value should be negotiated during initialization of this VCR.

Actual Outstanding Services Receiving (ActualMaxRCC)

This attribute specifies the maximum number of outstanding confirmed services allowed in this station as the responder side of this VCR. Its value should be negotiated during initialization of this VCR.

6.3.3.2.1.3 Services

Initiate

This optional service is used to open the VCR.

Abort

This optional service is used to close the VCR.

Reject

This optional service is used to reject a service primitive submitted to the VCR.

6.3.3.2.2 The transaction object class specification

6.3.3.2.2.1 Formal model

A Transaction Object should be created upon receipt of a confirmed service indication primitive. The Transaction Object should be uniquely related to the corresponding service primitive. The Transaction Object should be deleted after the corresponding response primitive has been sent. The Transaction Object should be uniquely identified by the combination of a protocol-created InstanceID and Static VCR ID.

The maximum number of Transaction Objects for each VCR of the server should be specified by the Max Outstanding Services Receiving attribute in the VCR entry. The following services act on the Transaction Object:

ASE:			**Context Management ASE**
CLASS:			**Transaction Object**
CLASS ID:			**—**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	instanceID & VCR ID
2	(m)	Attribute:	Confirmed service indication

6.3.3.2.2.2 Attributes

InstanceID & VCR ID

InstanceID

This attribute should identify the Transaction Object within the VCR.

VCR ID

This attribute should identify the VCR on which the Transaction Object is used.

Confirmed service indication

Service identifier and argument of the pending service.

6.3.3.2.2.3 Services

Initiate

This optional service is used to open the VCR.

Abort

This optional service is used to abruptly close the VCR.

Reject

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

6.3.3.2.2.4 State machine

6.3.3.2.2.4.1 State machine description

The state machine is shown in Figure 6.

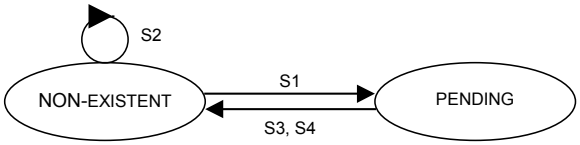


Figure 6 – Transaction object state machine

NON-EXISTENT

The Transaction Object does not exist.

PENDING

A confirmed service indication primitive has been received and the corresponding response primitive has not yet been sent.

6.3.3.2.2.4.2 State transitions

Table 18 specifies the state transitions related to transaction objects.

Table 18 – Transaction object state transitions

#	Current State	Events	Actions	Next State
S1	NON_EXISTENT	Confirmed service.ind && OSCR < Act MaxRCC	(no actions taken)	PENDING
S2	NON_EXISTENT	Confirmed service.ind && OSCR = Act MaxRCC	(no actions taken)	NON_EXISTENT
S3	PENDING	Confirmed service.rsp	(no actions taken)	NON_EXISTENT
S4	PENDING	Abort.ind	(no actions taken)	NON_EXISTENT

6.3.3.3 Context management ASE service specification

6.3.3.3.1 Supported services

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

– Initiate
– Abort
– Reject

6.3.3.3.2 Initiate service

6.3.3.3.2.1 Service overview

This service should be used to establish a connection between two communication partners and to exchange information regarding the supported services, the supported options, the maximum PDU length and the current version of the OD.

When the type of a VCR is such that FMS Initiate PDU is not exchanged, the request is locally confirmed and no information on the communication partner is available in confirmation.

6.3.3.3.2.2 Service primitives

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

Table 19 – Initiate service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Version OD Calling	M	M (=)		
Profile Number Calling	M	M (=)		
Access Protection Supported Calling	M	M (=)		
Password Calling	M	M (=)		
Access Groups Calling	M	M (=)		
Destination DL-Address	C			
Result(+)			S	S (=)
Version OD Called			M	M (=)
Profile Number Called			M	M (=)
Access Protection Supported Called			M	M (=)
Password Called			M	M (=)
Access Groups Called			M	M (=)
Result(-)			S	S (=)
Error Code			M	M (=)
Max APDU Length Sending Called				C
Max APDU Length Receiving Called				C
Features Supported Called				C

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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Version OD Calling

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

Profile Number Calling

This parameter specifies the value of the client's Profile Number attribute if it has one. If it does not, its value is null.

Access Protection Supported Calling

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

Password Calling

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

Access Groups Calling

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

Destination DL-Address

This parameter exists only when the corresponding AR supports it and the Remote Address Configuration Type is FREE. It gives the remote address to which the connection should be established.

Result(+)

This selection type parameter indicates that the service request succeeded.

Version OD Called

This parameter specifies the version of the server's OD.

Profile Number Called

This parameter specifies the value of the server's Profile Number attribute.

Access Protection Supported Called

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

Password Called

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

Access Groups Called

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

Result(-)

This selection type parameter indicates that the service request failed.

Error Code

This parameter should provide the reason for the failure as specified in Table 20.

Table 20 – Failure reasons

Reason	Meaning
Max APDU Size Insufficient	The maximum APDU length is not sufficient for the communication.
Feature Not Supported	The requested service or option is not supported by the server.
User Initiate Denied	The user refuses to establish the connection.
Version Object Definition incompatible	The called and calling versions of the Object Definitions are not compatible.
Password Error	There is already a VCR established with the same password, or the password is not valid.
Profile number incompatible	The client's profile number is not supported by the server.
Other	Reason other than any of those identified above.

Max PDU Length Sending Called

This parameter specifies the maximum length of the APDU to be sent by the server on this VCR.

Max PDU Length Receiving Called

This parameter specifies the maximum length of APDUs to be received by the server on this VCR.

Features Supported Called

This parameter identifies the optional services and options supported by the server on this VCR.

6.3.3.3.2.3 Service procedure

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

6.3.3.3 Abort service

6.3.3.3.1 Service overview

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

6.3.3.3.2 Service primitives

The service parameters for this service are shown in Table 21.

Table 21 – Abort service parameters

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

Argument

This parameter carries the parameters of the service invocation.

Locally Generated

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

The value false is not allowed if the Abort Identifier parameter has the value APO ASE and the Reason Code parameter has the value VCR Error.

Abort Identifier

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

USER

APO ASE

AR ASE

DLL

Reason Code

This parameter should specify the reason for the termination.

If the Abort Identifier parameter has the value USER, the values listed in Table 22 should apply.

Table 22 – User abort reasons

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

If the Abort Identifier parameter has the value APO ASE, the values listed in Table 23 should apply.

Table 23 – APO ASE abort reasons

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

If the Abort Identifier parameter has the value AR ASE or DLL, the Reason Code parameter should be supplied by the AR ASE.

Abort Detail

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

6.3.3.3.3 Service procedure

The Unconfirmed Service Procedure specified in Clause 4 applies to this service.

6.3.3.4 Reject service

6.3.3.4.1 Service overview

This service is used to inform the remote endpoint that a protocol error has been detected. It is generated by the AP ASE if an APDU has been received with an invalid service type code. When another APO ASE detects a protocol error, it internally requests the AP ASE to generate a Reject PDU. This service is not used to indicate that a service error has been detected by the user.

6.3.3.4.2 Service primitives

The service parameters for this service are shown in Table 24.

Table 24 – Reject service parameters

Parameter name	Ind
Argument	
VCR	M (=)
Detected Here	M
Original InstanceID	U
Reject APDU Type	M
Reject Code	M

Argument

The argument contains the parameters of the service request.

Detected Here

This parameter indicates if the error has been detected locally (TRUE). The value FALSE is allowed only if the Reject PDU Type parameter has the value Confirmed-Response-PDU and the Reject Code parameter has the value PDU Size.

Original InstanceID

This parameter specifies the InstanceID of the rejected APDU, if it has one.

Reject APDU Type

This parameter should indicate the type of the rejected PDU. The permissible values are as follows:

Confirmed-Request-PDU

Confirmed-Response-PDU

Unconfirmed-PDU

Unknown type of PDU

Reject Code

This parameter indicates the reason for rejection as listed in Table 25.

Table 25 – Reject APDU reasons

Reason	Meaning
Invoke-ID-Exists	Confirmed service.req received from the user and InstanceID already exists
Max-Services-Overflow.	Confirmed service.req received from the user and OSCS = ActualMaxSCC
Feature-Not-Supported-Connection-Oriented	Service.req received from the user and service or option not supported as a client (see Features Supported attribute in the VCR)
Feature-Not-Supported-Connectionless.	Unconfirmed service.req received from the user and service or option not supported as a client (see Features Supported attribute in the VCR) or confirmed service.req received from the user
APDU-Size	APDU length exceeds maximum APDU length
User-Error-Connectionless	Improper or faulty service primitive received from the user
Other	Reason other than any of those identified above

6.3.3.3.4.3 Service procedure

If any APO ASE receives an APDU that contains one of the above errors, the AP ASE delivers a Reject indication primitive.

If any APO ASE receives a confirmed request primitive that contains one of the above errors, the AP ASE delivers a Reject indication primitive.

If any APO ASE receives a confirmed response primitive that contains one of the above errors, the AP ASE builds a Reject Request APDU and sends it on the specified AR.

Upon receipt of the Reject Request APDU, the receiving AP ASE delivers an AR-Reject.indication primitive to the requesting user.

6.3.3.4 Context management ASE tests at connection establishment

6.3.3.4.1 Context test in the AE

Upon receipt of an INITIATE-REQ_PDU the called AE should check whether the context of the communication partner (remote Context) is compatible with its own context (local context) as defined for this connection in this VCR, as shown in Table 26. The local context is assumed to be correct. The compatibility of the local with the remote context is defined by the following table. In this table, meaningless fields are left blank (e.g. the combination of local context "Max APDU Sending " with remote context "Features Supported"). Those combinations should not be checked.

Table 26 – Compatibility of the local context to the remote context

Remote Content			Local Content: Max APDU Sending	Local Content: Max APDU Receiving	Local Content: Feature Supported Req [n] 0	Local Content: Feature Supported Req [n] 1	Local Content: Feature Supported Ind [n] 0	Local Content: Feature Supported Ind [n] 1
Max APDU Sending				≥				
Max APDU Receiving			≤					
Features Supported	.req [n]	0					X	X
	.req [n]	1					-	X
	.ind [n]	0			X	-		
	.ind [n]	1			X	X		

Explanation:

.req 0	:	Feature is not used as Client
.req 1	:	Feature is used as Client
.ind 0	:	Feature is not supported as Server
.ind 1	:	Feature is supported as Server
≤	:	local value smaller than or equal remote value
≥	:	local value larger than or equal remote value
X	:	compatible
–	:	not compatible (error case)
[n]	:	0 to (23+8x); without extension ==> x= 0; with extension ==> x= 1,2, ...

The length of the Features-Supported is not limited. So it can happen, that the bitstring length of the Features-Supported is different between the bitstring length of the Features-Supported from the communication partners. To check the context it is necessary to approximate the both bitstring length, which should be compared. The rule is, that the bitstring with the shorter length has to be extended with bits of the value 0, until both bitstrings are equal. Note that the bitstring of the Features Supported consists of a bitstring for the primitive of client functions and a bitstring for the primitive of server functions. The extension has to be symmetrical between the bitstring for the primitive of client functions and the bitstring for the primitive of server functions. This is shown in Figure 7.

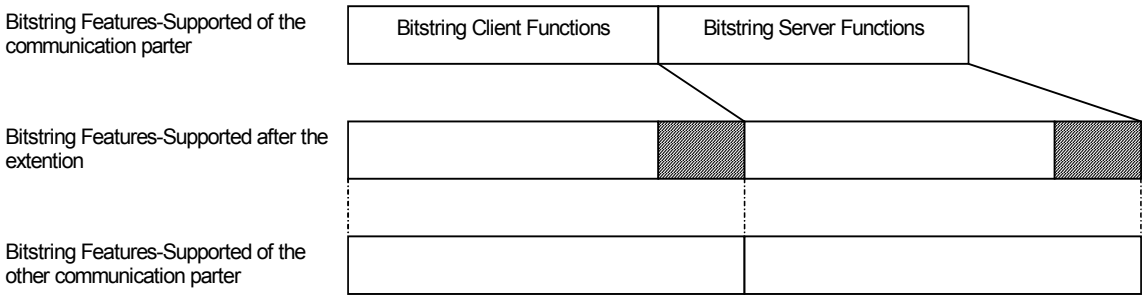


Figure 7 – Context test of two features-supported with different bitstring length

MaxRCC and MaxSCC of initiating endpoint are shown in INITIATE-REQ PDU. Upon reception of this PDU, the called AE should store the smaller one of MaxRCC in the PDU and its own MaxSCC to its ActualMaxSCC. It should also store the smaller one of MaxSCC in the PDU and its own MaxRCC to its ActualMaxRCC. The called AE should return the resultant ActualMaxRCC and ActualMaxSCC in the MaxRCC and MaxSCC fields of INITIATE-RSP_PDU. Upon reception of this PDU, the initiating FMS should store them in its own Act MaxSCC and ActualMaxRCC, respectively. If the ActualMaxRCC is greater than MaxRCC or ActualMaxSCC is greater than MaxSCC, the initiating AE should abort this connection.

6.3.3.4.2 Context test in the user

The following context shall be checked by the AE user.

- Password Test. If the user supports access with password, it should check if the password is unambiguous. That means, if there is a password, it should be different from the passwords of all other VCRs.
- If the password is not unambiguous and the user is the server of the Initiate service, it should issue an Initiate.rsp primitive with the Result(-) parameter and with Error Code "Password Error".
- If the password is not unambiguous and the user is the client of the Initiate service, it should release the connection with the Abort service, with Reason Code "Password Error".
- Test of Version OD. If the user has a remote OD for this connection, it may check if the received parameter "Version OD" is compatible with the attribute "Version OD" of the associated remote OD.
- If they are incompatible and the user is the server of the Initiate service, it may issue an Initiate.rsp primitive with the Result(-) parameter and with Error Code "Version OD incompatible".
- If they are incompatible and the user is the client of the Initiate service, it may release the connection with the Abort service, with Reason Code "Version OD incompatible".
- Test of Profile Number. The user may check if the received parameter "Profile Number" is compatible with the attribute "Profile Number" of the VFD.
- If they are incompatible and the user is the server of the Initiate service, it may issue an Initiate.rsp primitive with the Result(-) parameter and with Error Code "Profile Number incompatible".
- If they are incompatible and the user is the client of the Initiate service, it may release the connection with the Abort service, with Reason Code "Profile Number incompatible".

6.3.4 Application relationship ASE

6.3.4.1 Overview

The AR ASE supports three types of ARs, all derived from the top level AREP class shown below.

6.3.4.2 Application relationship endpoint class specification

6.3.4.2.1 Formal model

ASE:			**AR ASE**
CLASS:			**AR ENDPOINT**
CLASS ID:			
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
SERVICES:			
1	(m)	OpsService:	AR-Associate
2	(m)	OpsService:	AR-Abort
3	(o)	OpsService:	Confirmed Send
4	(o)	OpsService:	Unconfirmed Send
5	(o)	OpsService:	Compel
6	(o)	OpsService:	Get-Buffered-Message
7	(o)	OpsService:	AR-Status

6.3.4.2.2 Attributes

Numeric Identifier

Numeric Identifier of the AREP.

6.3.4.2.3 Services

AR-Associate

This service is used to establish an AR.

AR-Abort

This service is used to abruptly de-establish an AR.

Confirmed Send

This service is used to send a confirmed service on the specified AR.

Unconfirmed Send

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

Compel

This optional service is used to evoke acyclic communication out on a BNU AREP.

Get-Buffered-Message

This optional service is used to retrieve the current contents of the associated buffer.

AR-Status

This optional service is used to report status of the AR.

6.3.4.3 Application relationship ASE service specifications

6.3.4.3.1 Unconfirmed send service

6.3.4.3.1.1 Service overview

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

6.3.4.3.1.2 Service primitives

The service parameters for this service are shown in Table 27.

Table 27 – Unconfirmed send service parameters

Parameter name	Req	Ind
Argument		
AREP	M	M
Remote DLSAP Address	C	C (=)
Duplicate FAL PDU Body		C
FAL APDU Body	M	M (=)
Local Timeliness		C
Remote Timeliness		C

Argument

The argument contains the parameters of the service request.

Remote DLSAP Address

This conditional parameter is present if the ConfigurationType attribute of the associated AREP is Free. If it is present, this parameter carries the DLSAP address associated with the remote AREP. If the ConfigurationType attribute is Linked, this parameter is not present.

Duplicate FAL PDU Body

This conditional parameter is present if the DuplicatePduDetectionSupported attribute is True. If it is present, it indicates whether or not the FAL APDU Body is a duplicate of the FAL APDU Body that has been received and delivered in a previous service indication.

This parameter takes a value of True if the same FAL APDU Bodies have been received successively. Otherwise, it takes a value of False.

FAL APDU Body

This parameter specifies an unconfirmed service request APDU.

Local Timeliness

This conditional parameter, if it is present and True, indicates that the FAL APDU Body parameter has met the receiving timeliness criteria defined for the DLL. If its value is False, at least one of the timeliness criteria has not been met.

This parameter is present if the value of theSubscriberD1TimelinessClass attribute is other than None and is not present if it is None.

Remote Timeliness

This conditional parameter, if it is present and True, indicates that the FAL APDU Body parameter has met the Publisher's and transmitting DL's timeliness criteria. If its value is False, at least one of the timeliness criteria has not been met.

This parameter is present if the value of thePublisherD1TimelinessClass attribute is other than None and is not present if it is None.

6.3.4.3.2 Confirmed send service

6.3.4.3.2.1 Service overview

This service is used to send a confirmed service on the specified AR.

6.3.4.3.2.2 Service primitives

The service parameters for this service are shown in Table 28.

Table 28 – Confirmed send service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
FAL APDU Body	M	M (=)		
Result				
FAL APDU Body			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. See 1.2.

Argument

The argument contains the parameters of the service request.

FAL APDU Body

This parameter specifies an confirmed service request APDU.

Result(+)
FAL APDU Body

This parameter specifies an confirmed service response APDU.

6.3.4.3.3 AR-abort service

6.3.4.3.3.1 Service overview

This service is used to abruptly de-establish an AR.

6.3.4.3.3.2 Service primitives

The service parameters for this service are shown in Table 29.

Table 29 – AR-Abort service parameters

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

Argument

The argument contains the parameters of the service request.

Locally Generated

This parameter indicates whether the Abort service was initiated at the local or remote AREP.

Identifier

This parameter indicates whether the Abort service was initiated at the local or remote AREP.

NOTE In cases when the Abort.request or Abort.indication is generated by the AREP, the Additional Detail parameter does not exist. If the Abort.indication is generated by the local AREP, the Identifier and Reason Code parameters are set locally.

Reason Code

This parameter specifies the reason for the AR abort.

Additional Detail

This optional parameter specifies additional information about the AR release.

6.3.4.3.4 Compel service

6.3.4.3.4.1 Service overview

This optional service is used to evoke acyclic communication out on a BNU AREP.

6.3.4.3.4.2 Service primitives

The service parameters for this service are shown in Table 30.

Table 30 – Compel service parameters

Parameter name	Req	Cnf
Argument		
AREP	M	
Result		
Status		M

Argument

The argument contains the parameters of the service request.

Result
Status

This parameter indicates the result of the service request. The following three status codes provided by the DLL are defined:

success

failure – inappropriate request

failure – reason unspecified

6.3.4.3.5 Get buffered message service

6.3.4.3.5.1 Service overview

This optional service is used to retrieve the current contents of the associated buffer.

6.3.4.3.5.2 Service primitives

The service parameters for this service are shown in Table 31.

Table 31 – Get buffered message service parameters

Parameter name	Req	Cnf
Argument		
AREP	M	
Result(+)		S
Duplicate APDU Body		C
FAL PDU Body		M
Local timeliness		C
Remote timeliness		C
Result (-)		S

Argument

The argument contains the parameters of the service request.

Result(+)
Duplicate FAL PDU Body

This conditional parameter is present if the DuplicatePduDetectionSupported attribute is True. If it is present, it indicates whether or not the FAL APDU Body is a duplicate of the FAL APDU Body that has been received and delivered in a previous service indication.

This parameter takes a value of True if the same FAL APDU Bodies have been received successively. Otherwise, it takes a value of False.

FAL APDU Body

This parameter specifies an unconfirmed service request APDU.

Local Timeliness

This conditional parameter, if it is present and True, indicates that the FAL APDU Body parameter has met the receiving timeliness criteria defined for the DLL. If its value is False, at least one of the timeliness criteria has not been met.

This parameter is present if the value of theSubscriberD1TimelinessClass attribute is other than None and is not present if it is None.

Remote Timeliness

This conditional parameter, if it is present and True, indicates that the FAL APDU Body parameter has met the Publisher's and transmitting DL's timeliness criteria. If its value is False, at least one of the timeliness criteria has not been met.

This parameter is present if the value of thePublisherD1TimelinessClass attribute is other than None and is not present if it is None.

Result(-)

This parameter indicates that the service request failed. When this parameter is returned, it means that the buffer attached to the specified AREP does not contain any FAS-PDUs.

6.3.4.3.6 AR-status service

6.3.4.3.6.1 Service overview

This optional service is used to report status of the AR.

6.3.4.3.6.2 Service primitives

The service parameters for this service are shown in Table 32.

Table 32 – AR-Status service parameters

Parameter name	Ind
Argument	
AREP	M
Status code	M

Argument

The argument contains the parameters of the service request.

Status Code

This parameter specifies the event which is reported from the DLL. The following event is specified:

Buffer-Sent: The DLL reported that the buffer content had just been sent.

6.3.5 Variable ASE

6.3.5.1 Overview

In the fieldbus environment, application processes contain data that remote applications are able to read and write. The variable ASE defines the network visible attributes of application data and provides a set of services used to read, write, and report their values. Common FAL management services are used to create and delete variable objects and to access their attributes.

6.3.5.2 Variable model class specifications

6.3.5.2.1 Simple variable class specification

6.3.5.2.1.1 Formal model

The Simple Variable Object represents a single, simple variable which is characterized by a defined Data type. The Object Description of the Variable Access Object Simple Variable is stored statically in the Object Dictionary (S-OD). The mapping of a Simple Variable to a real Simple Variable, which exists in the application system, is defined by the object description of the Simple Variable Object. Only one single Data type of the set of configured Data types is allowed in the Object Description of the Simple Variable Object.

ASE:			**VARIABLE ASE**
CLASS:			**SIMPLE VARIABLE**
CLASS ID:			**7**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Data type Index
3	(m)	Attribute:	Length
4	(m)	Attribute:	Password
5	(m)	Attribute:	Access Groups
6	(m)	Attribute:	Access Rights
7	(m)	Attribute:	Local Address
8	(m)	Attribute:	Extension
SERVICES:			
1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
3	(o)	OpsService:	Information Report

6.3.5.2.1.2 Attributes

Numeric Identifier

Identifies an instance of this object class.

Data type Index

Logical address of the corresponding Data type in the Static Type Dictionary (ST-OD).

Length

Length in octets of the data.

Password

This attribute specifies the password for the Access Rights.

Access Groups

This attribute relates the object to specific Access Groups as shown in Table 33. The object is a group member when the corresponding bit is set.

Table 33 – Simple variable access group membership

Bit	Meaning
7	Access Group 1
6	Access Group 2
5	Access Group 3
4	Access Group 4
3	Access Group 5
2	Access Group 6
1	Access Group 7
0	Access Group 8

Access Rights

This attribute specifies information about the Access Rights as shown in Table 34. The specific Access Right exists when the corresponding bit is set.

Table 34 – Simple variable access rights membership

Bit	Name	Meaning
7	R	Right to Read for the registered Password
6	W	Right to Write for the registered Password
3	Rg	Right to Read for Access Groups
2	Wg	Right to Write for Access Groups
15	Ra	Right to Read for all Communication Partners
14	Wa	Right to Write for all Communication Partners

Local Address

This attribute is a system specific address of the real object. It serves the internal addressing of the object. If no mapping of this kind is performed, this attribute should have the value FFFFFFFF hex.

Extension

This attribute specifies profile specific information.

6.3.5.2.1.3 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 service permits the VFD to send the value of a variable.

6.3.5.2.2 Array variable class specification

6.3.5.2.2.1 Formal model

The Array Object is used to define a constructed variable in which all elements have the same Data type and length.

The Object Description of the Variable Access Object Array is stored statically in the Object Dictionary (S-OD). The mapping of a Array Variable to a real array, which exists in the application system, is described by the object description of this object. The object description for Array Objects specifies the number of elements in the array, the Data type and Length of the elements.

The Array Object may be accessed completely, or element by element using subindexes. Subindex 1 accesses the first element of the object.

ASE:			**VARIABLE ASE**
CLASS:			**ARRAY VARIABLE**
CLASS ID:			**8**
PARENT CLASS:			**SIMPLE VARIABLE**
ATTRIBUTES:			
1	(m)	Attribute:	Number of Elements

6.3.5.2.2.2 Attributes

Number of Elements

States how many elements are contained in the Array.

6.3.5.2.3 Record variable class specification

6.3.5.2.3.1 Formal model

The Record Object consists of a collection of Simple Variables of different Data types.

The Object Description of the Variable Access Object Record is stored dynamically in the Object Description (S-OD).

The mapping of a Record Variable to a real record, which exists in the application system, is described by the object description of this object. The object description states the relation of the object to a configured Data type Structure Description.

The Record Object may be accessed completely or element-wise. If the Object should be accessed completely the index should be used for the service. If an element of the object should be accessed then the subindex should be used together with the index for the service. Subindex 1 accesses the first element of the object.

ASE:			**VARIABLE ASE**
CLASS:			**RECORD VARIABLE**
CLASS ID:			**9**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Data type Index
3	(m)	Attribute:	Password
4	(m)	Attribute:	Access Groups
5	(m)	Attribute:	Access Rights
6	(m)	Attribute:	List of Local Address
6.1	(m)	Attribute:	Local Address
7	(m)	Attribute:	Extension
SERVICES:			
1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
3	(o)	OpsService:	Information Report

6.3.5.2.3.2 Attributes

Numeric Identifier

Identifies an instance of this object class.

Data type Index

Logical address of the corresponding Data type in the Static Type Dictionary (ST-OD).

Length

Length in octets of the data.

Password

This attribute specifies the password for the Access Rights.

Access Groups

This attribute relates the object to specific Access Groups as shown in

Table 35. The object is a group member when the corresponding bit is set.

Table 35 – Array variable access group membership

Bit	Meaning
7	Access Group 1
6	Access Group 2
5	Access Group 3
4	Access Group 4
3	Access Group 5
2	Access Group 6
1	Access Group 7
0	Access Group 8

Access Rights

This attribute specifies information about the Access Rights as shown in Table 36. The specific Access Right exists when the corresponding bit is set.

Table 36 – Array variable access rights membership

Bit	Name	Meaning
7	R	Right to Read for the registered Password
6	W	Right to Write for the registered Password
3	Rg	Right to Read for Access Groups
2	Wg	Right to Write for Access Groups
15	Ra	Right to Read for all Communication Partners
14	Wa	Right to Write for all Communication Partners

List of Local Address

This attribute is a list of addresses, one for each element in the record. The order in the list is defined by the order of the elements in the record.

Local Address

This attribute is a system specific address of the real object. It serves the internal addressing of the object. . If the real objects are stored sequentially without gaps, then only the Local Address of the first element is given. If no mapping of this kind is performed, this attribute should have the value FFFFFFFF hex.

Extension

This attribute specifies profile specific information.

6.3.5.2.3.3 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 service permits the VFD to send the value of a variable.

6.3.5.2.4 Variable list class specification

6.3.5.2.4.1 Formal model

The Variable List Object consists of a collection of Simple, Array, and Record Variable Objects.

The Object Description of the Variable Access Object Variable List is stored dynamically in the Object Dictionary (DV-OD). This Object Description contains an index list of the contained variable objects from the S-OD.

A Variable List may be created and deleted with the DefineVariableList and DeleteVariableList services.

The Access Rights should be declared for the DefineVariableList service. The Access Rights to the single objects are checked on creation (DefineVariableList) of a Variable List.

The Variable List Object is only created if the requested Access Rights do not exceed the Access Rights to the single objects.

If the same Variable List exists with the requested Access Rights, no new object is created but the Logical Address of the existing object is given to the client. In the other case the Variable List is created with the requested Access Rights.

Only the authorized client with the proper Access Rights may delete a Variable List Object (DeleteVariableList).

If the Object Dictionary is freshly loaded all Variable List Objects are deleted.

ASE:			**VARIABLE ASE**
CLASS:			**VARIABLE LIST**
CLASS ID:			**10**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Number of Elements
3	(m)	Attribute:	Password
4	(m)	Attribute:	Access Groups
5	(m)	Attribute:	Access Rights
6	(c)	Attribute:	Deletable
7	(m)	Attribute:	List Of Element Index
7.1	(m)	Attribute:	Index
8	(m)	Attribute:	Extension
SERVICES:			
1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
3	(o)	OpsService:	Information Report
4	(o)	OpsService:	Define Variable List
5	(o)	OpsService:	Delete Variable List

6.3.5.2.4.2 Attributes

Numeric Identifier

Identifies an instance of this object class.

Number Of Elements

This parameter states the number of Variable Objects in the Variable List.

Password

This attribute specifies the password for the Access Rights.

Access Groups

This attribute relates the object to specific Access Groups as shown in Table 37. The object is a group member when the corresponding bit is set.

Table 37 – Variable list access group membership

Bit	Meaning
7	Access Group 1
6	Access Group 2
5	Access Group 3
4	Access Group 4
3	Access Group 5
2	Access Group 6
1	Access Group 7
0	Access Group 8

Access Rights

This attribute specifies information about the Access Rights as shown in Table 38. The specific Access Right exists when the corresponding bit is set.

Table 38 – Variable list access rights membership

Bit	Name	Meaning
7	R	Right to Read for the registered Password
6	W	Right to Write for the registered Password
5	D	Right to Delete for the registered Password
3	Rg	Right to Read for Access Groups
2	Wg	Right to Write for Access Groups
1	Dg	Right to Delete for Access Groups
15	Ra	Right to Read for all Communication Partners
14	Wa	Right to Write for all Communication Partners
13	Da	Right to Delete for all Communication Partners

Deletable

This attribute indicates if the Variable List Object may be deleted (true) or not (false) using the DeleteVariableList service.

List of Element Index

This attribute specifies the indices of the Variable Access Objects which are combined to this Variable List.

Extension

This attribute specifies profile specific information.

6.3.5.2.4.3 Services

Read

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

Write

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

Information Report

This service permits the VFD to send the value of a variable list.

Define Variable List

This service creates a variable list object.

Delete Variable List

This service deletes a variable list object.

6.3.5.2.5 Variable data type class specification

6.3.5.2.5.1 Formal model

The Data type defines the syntax, the range of the variables and their presentation inside the communication system. The Data type of an object is indicated by a corresponding attribute in the Object Description. Type definitions may not be made remotely for reasons of efficiency. They have a fixed configuration in the Static Type Dictionary (ST-OD).

This class defines free configurable Data types and Standard Data types. They are stored beginning with Index 1 in the Static Type Dictionary (ST-OD). Standard Data types which are not used are marked as not configured in the ST-OD.

ASE:			**VARIABLE ASE**
CLASS:			**DATA TYPE**
CLASS ID:			
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Description
2.1	(m)	Attribute:	Symbolic Name Length
2.2	(m)	Attribute:	Symbolic Name

6.3.5.2.5.2 Attributes

Numeric Identifier

Identifies an instance of this object class.

Description

This attribute specifies a textual description of the Data type. It consists of the Symbolic Name Length and the Symbolic Name.

Symbolic Name Length

This attribute specifies the length of the symbol. The value 0 means that no symbol is used.

Symbolic Name

This attribute specifies a visible string of 0 to 32 octets as a symbolic description.

6.3.5.2.5.3 Services

None.

6.3.5.2.6 Variable data type structure class specification

6.3.5.2.6.1 Formal model

The Data type Structure Object characterizes the size and the structure of records.

The Object Description of the Data type Structure Object has a fixed configuration in the Static Type Dictionary. It contains a list of elements, consisting of Data type and Length of the corresponding record element. As Data types for the single elements only Data types in the set of configured Data types of the Static Type Dictionary are allowed.

ASE:			**VARIABLE ASE**
CLASS:			**DATA TYPE STRUCTURE**
CLASS ID:			
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Number Of Elements
3	(m)	Attribute:	List Of Elements
3.1	(m)	Attribute:	Data type Index
3.2	(m)	Attribute:	Length

6.3.5.2.6.2 Attributes

Numeric Identifier

Identifies an instance of this object class.

Number Of Elements

Number of elements of the structured Data type.

List Of Elements

This attribute specifies a list of elements, consisting of Data type and Length of the corresponding record element.

Data type Index

Index in the ST-OD of the Data type of element n.

Length

Length of the element n in octets.

6.3.5.2.6.3 Services

None.

6.3.5.3 Variable ASE service specification

6.3.5.3.1 Supported services

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

Read

Write

Information Report

Define Variable List

Delete Variable List

6.3.5.3.2 Read service

6.3.5.3.2.1 Service overview

This service permits a client to read the value of a variable or a variable list object.

6.3.5.3.2.2 Service primitives

The service parameters for this service are shown in Table 39.

Table 39 – Read service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Variable Index	M	M (=)		
Variable Subindex	U	U (=)		
Result (+)			S	S (=)
Data			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the parameters of the service request.

Variable Index

This parameter is the OD index of the variable or variable list.

Variable Subindex

This optional parameter identifies an individual element in an array or record variable by its position within the variable, starting at 1.

Result(+)
Data

This parameter specifies the value of the variable or variable list.

6.3.5.3.3 Write service

6.3.5.3.3.1 Service overview

This service permits a client to write the value of a variable or a variable list object.

6.3.5.3.3.2 Service primitives

The service parameters for this service are shown in Table 40.

Table 40 – Write service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Variable Index	M	M (=)		
Variable Subindex	U	U (=)		
Data	U	U (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the parameters of the service request.

Variable Index

This parameter is the OD index of the variable or variable list.

Variable Subindex

This optional parameter identifies an individual element in an array or record variable by its position within the variable, starting at 1.

Data

This parameter specifies the value of the variable or variable list.

6.3.5.3.4 Information report service

6.3.5.3.4.1 Service overview

This service permits the VFD to send the value of a variable or a variable list object.

6.3.5.3.4.2 Service primitives

The service parameters for this service are shown in Table 41.

Table 41 – Information report service parameters

Parameter name	Req	Ind
Argument		
VCR	M	M
Destination DL-Address	C	
Source DL-Address		C
Duplicated APDU Body		C
Variable Index	M	M (=)
Variable Subindex	U	U (=)
Data	U	U (=)
On Change	U	U (=)

Argument

This parameter carries the parameters of the service invocation.

Destination DL-Address

This parameter exists only when type of the corresponding VCR is QUU and the remote partner has FREE endpoint. It gives remote address to which the requested Data should be sent.

Source DL-Address

This parameter exists only when type of the corresponding VCR is QUU and has FREE endpoint. It indicates address from which the indicated data was sent.

Duplicated APDU Body

This parameter exists only when FAS provides it for BNU VCR and indicates whether or not the data parameter has been received and delivered in a previous service indication.

Variable Index

This parameter is the OD index of the variable or variable list.

Variable Subindex

This optional parameter identifies an individual element in an array or record variable by its position within the variable, starting at 1.

Data

This parameter specifies the value of the variable or variable list.

On Change

This optional Boolean parameter indicates, when TRUE, that the variable value is being reported because it has changed, or because it has not changed within an externally specified time period.

6.3.5.3.5 Define variable list service

A Variable List Object is created at the communication partner using this service. The Number Of Elements attribute is calculated automatically by the server for the Define Variable List service. The client user should ensure that the data of the Variable List Service may be transmitted in one single APDU. The Deletable attribute of a Variable List which is created by the Define Variable List service is set to the value TRUE.

6.3.5.3.5.1 Service overview

This service creates a variable list object.

6.3.5.3.5.2 Service primitives

The service parameters for this service are shown in Table 42.

Table 42 – Define variable list service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Password	M	M (=)		
Access Groups	M	M (=)		
Access Rights	M	M (=)		
List of Variable Indexes	M	M (=)		
Extension	U	U (=)		
Result (+)			S	S (=)
OD Index			M	M (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.				

Argument

This parameter carries the parameters of the service invocation.

Password

This parameter states to which value the Password attribute is to be set.

Access Groups

This parameter states to which value the attribute Access Groups is to be set.

Access Rights

This parameter states to which value the attribute Access Rights is to be set.

List of Variable Indexes

This parameter specifies the OD indexes of the variables which are combined in this Variable List.

Extension

This attribute specifies profile specific information to be set.

Result(+)

The Result(+) parameter should indicate that the service was completed successfully.

OD Index

OD Index of the newly created Variable List object.

Result(-)

The Result(-) should indicate that the service request was not completed successfully.

6.3.5.3.5.3 Service procedure

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

6.3.5.3.6 Delete variable list service

6.3.5.3.6.1 Service overview

This service deletes a variable list object.

A Variable List Object is deleted with this service if there is an Access Right for this Object. Only Variable List objects having the value true for the Deletable attribute may be deleted.

6.3.5.3.6.2 Service primitives

The service parameters for this service are shown in Table 43.

Table 43 – Delete variable list service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
OD Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

OD Index

This parameter identifies the variable list by its OD Index.

Result(+)

The Result(+) parameter should indicate that the service was completed successfully.

Result(-)

The Result(-) should indicate that the service request was not completed successfully.

6.3.5.3.6.3 Service procedure

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

6.3.6 Event ASE

6.3.6.1 Overview

The Event serves for sending an important message from one device to another (or in multicast mode to selected devices) as shown in Figure 8. The detection of the conditions that cause an event is responsibility of the user. The application program invokes the Event

Notification service when the conditions are met. The monitoring and checking of the (optional) acknowledgment is also the responsibility of the user.

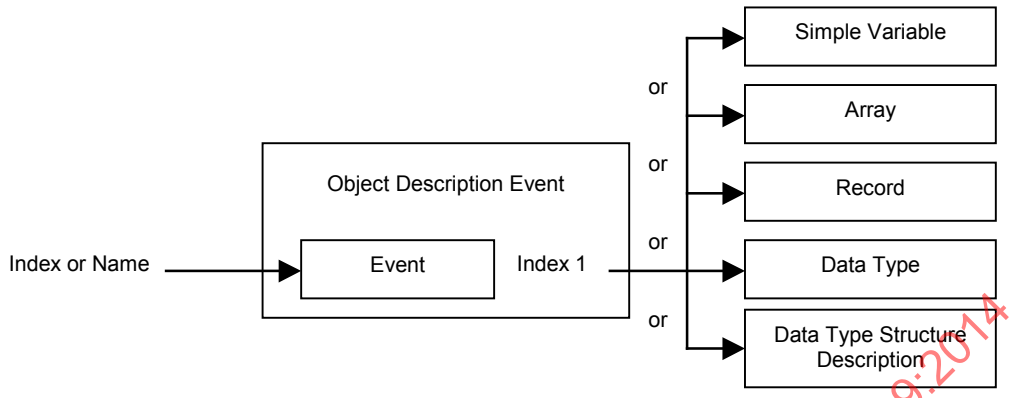


Figure 8 – Overview of event

A user sends an event notification with the EventNotification service.

The user transmits with the EventNotification service a counter value (Event Number) and data (Event Data), e.g. measurement value, status, units.

The management (increment) of the counter value is the responsibility of the user. The Event Notification may represent a collective alarm, where the data may contain information about which channel has triggered the collective alarm. Also in this case the logical operations for the triggering conditions and for the data are part of the application program.

The receiver of the Event Notification may acknowledge the Event-Notification by using the AcknowledgeEventNotification service.

The counter value is transmitted with the AcknowledgeEventNotification service. The counter value serves to reliably correlate the EventNotification and the AcknowledgeEventNotification. The monitoring and the checking of the acknowledgment is the responsibility of the user.

The receiver of the EventNotification may lock or unlock the EventNotification using the AlterEventConditionMonitoring service.

6.3.6.2 Event model class specifications

6.3.6.2.1 Class overview

This subclause provides formal class definitions for event objects. Each class definition specifies a type of event class as a combination of its attributes and the services defined for them.

6.3.6.2.2 Formal model

ASE:			**EVENT ASE**
CLASS:			**EVENT**
CLASS ID:			**4**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Attribute:	Numeric Identifier
2	(m)	Attribute:	Index of Event Data
3	(m)	Attribute:	Length
4	(m)	Attribute:	Password

5	(m)	Attribute:	Access Groups
6	(m)	Attribute:	Access Rights
7	(m)	Attribute:	Enabled
8	(m)	Attribute:	Extension

SERVICES:

1	(m)	OpsService:	Event Notification
2	(m)	OpsService:	Acknowledge Event Notification
3	(m)	OpsService:	Alter Event Condition Monitoring

6.3.6.2.3 Attributes

Numeric Identifier

Identifies an instance of this object class.

Index of Event Data

This attribute is the OD Index for event data associated with this event object. It may be the OD Index of a variable object or the OD index of the data type of the event data.

Length

If the Index Event Data points to a Data type then this attribute specifies the length in octets of the Event Data. This attribute is not significant otherwise.

Password

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

Access Groups

This attribute relates the object to specific Access Groups as shown in Table 44. The object is a group member when the corresponding bit is set.

Table 44 – Event access group membership

Bit	Meaning
7	Access Group 1
6	Access Group 2
5	Access Group 3
4	Access Group 4
3	Access Group 5
2	Access Group 6
1	Access Group 7
0	Access Group 8

Access Rights

This attribute specifies information about the Access Rights as shown in Table 45. The specific Access Right exists when the corresponding bit is set.

Table 45 – Event access rights membership

Bit	Name	Meaning
6	W	Right to Acknowledge for the registered Password
5	D	Right to Enable/Disable for the registered Password
2	Wg	Right to Acknowledge for Access Groups
1	Dg	Right to Enable/Disable for Access Groups
14	Wa	Right to Acknowledge for all Communication Partners
13	Da	Right to Enable/Disable for all Communication Partners

Enabled

This Boolean attribute indicates, when TRUE, that the event is enabled (unlocked).

Extension

This attribute specifies profile specific information.

6.3.6.2.4 Services

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

Event Notification

This service is provided to permit notifier object to report the occurrence of one or more events in a single service invocation.

Acknowledge Event Notification

This optional confirmed service is provided to permit a subscriber of an event to acknowledge receipt of an event message.

Enable Event Condition Monitoring

This optional service is provided to permit event reporting to be enabled for an event.

6.3.6.3 Event ASE service specifications

6.3.6.3.1 Supported services

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

- Event Notification
- Acknowledge Event
- Enable Event Condition Monitoring

6.3.6.3.2 Event notification

6.3.6.3.2.1 Service overview

This unconfirmed service is used to send an Event Notification, indicating that an event has been detected. This service is unconfirmed.

6.3.6.3.2.2 Service primitives

The service parameters for this service are shown in Table 46.

Table 46 – Event notification service parameters

Parameter name	Req	Ind
Argument		
VCR	M	M
Destination DL-Address	C	
Source DL-Address		C
Event Index	M	M (=)
Event Number	M	M (=)
Event Data	M	M (=)

Argument

The argument contains the parameters of the service request.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the destination address to which the requested Event Notification is to be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the source address from which the indicated Event Notification is to be sent.

Event Index

This parameter identifies the event being reported by this service using it OD index.

Event Number

This parameter reports the event count for the occurrence being reported.

Event Data

This parameter specifies the data. The data of the object should be mapped onto the parameter data according to the description of the Data types. The Index is stored as Index Event Data in the Object Description of the Event Object.

6.3.6.3.2.3 Service procedure

The Unconfirmed Service Procedure specified in Clause 4 applies to this service.

6.3.6.3.3 Acknowledge event notification

6.3.6.3.3.1 Service overview

This service allows the acknowledgment of a reported event.

6.3.6.3.3.2 Service primitives

The service parameters for this service are shown on Table 47.

Table 47 – Acknowledge event notification service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Event Index	M	M (=)		
Event Number	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the parameters of the service request.

Event Index

This parameter identifies the event object that is being acknowledged by its OD Index.

Event Number

This parameter reports the event count for the occurrence being reported.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.6.3.3.3 Service procedure

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

The association between the event identified in this service and the corresponding event object in the receiving AP is performed by the user.

6.3.6.3.4 Alter event condition monitoring

6.3.6.3.4.1 Service overview

This service allows the enabling or disabling of the Event Object.

6.3.6.3.4.2 Service primitives

The service parameters for this service are shown in Table 48.

Table 48 – Alter event condition monitoring service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Event Index	M	M (=)		
Enabled	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Event Index

This parameter identifies the event object by its OD index.

Enabled

This Boolean parameter, when TRUE, indicates that the event object is to be enabled, and when FALSE, that the event object is to be disabled. It is used to set or clear the attribute value of the same name.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.6.3.4.3 Service procedure

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

6.3.6.4 Event notification state machine

6.3.6.4.1 State machine description

6.3.6.4.1.1 States

The states are shown in Figure 9.

LOCKED

No Event Notifications are send in the LOCKED state (Enabled = false).

UNLOCKED

Event Notifications are sent normally in the UNLOCKED state (Enabled = true).

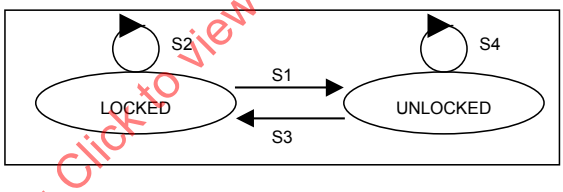


Figure 9 – Event state machine

6.3.6.4.1.2 State transitions

Table 49 specifies the state transitions related to events.

Table 49 – Event state transitions

#	Current State	Events	Actions	Next State
S1	LOCKED	AlterEventConditionMonitoring.ind && Enabled = true	AlterEventConditionMonitoring.rsp(+){ }	UNLOCKED
S2	LOCKED	AlterEventConditionMonitoring.ind && Enabled = false	AlterEventConditionMonitoring.rsp(+){ }	LOCKED
S3	UNLOCKED	AlterEventConditionMonitoring.ind && Enabled = false	AlterEventConditionMonitoring.rsp(+){ }	LOCKED
S4	UNLOCKED	AlterEventConditionMonitoring.ind && Enabled = true	AlterEventConditionMonitoring.rsp(+){ }	UNLOCKED

6.3.7 Domain ASE

6.3.7.1 Overview

A Domain represents an unstructured memory area whose contents may be uploaded (read) or downloaded (written). Unstructured in this context means that the memory area is represented only as an ordered sequence of octets. No other structure is apparent.

6.3.7.2 Domain model specification

6.3.7.2.1 Formal model

ASE:			**LOAD REGION ASE**
CLASS:			**LOAD REGION**
CLASS ID:			**2**
PARENT CLASS:			**TOP**
ATTRIBUTES:			
1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Max Octets
3	(m)	Attribute:	Password
4	(m)	Attribute:	Access Groups
5	(m)	Attribute:	Access Rights
6	(m)	Attribute:	Local Address
7	(m)	Attribute:	Domain State
8	(m)	Attribute:	Upload State
9	(m)	Attribute:	Counter
10	(m)	Attribute:	Extension
SERVICES:			
1	(m)	Ops Service:	Generic Download Services
2	(m)	Ops Service:	Download Services
3	(m)	Ops Service:	Upload Services

6.3.7.2.2 Attributes

Numeric Identifier

Identifies an instance of this object class.

Max Octets

This attribute specifies the max. number of octets of the Domain.

Password

This attribute should specify the password for the access rights.

Access Groups

This attribute should specify the object's membership in specific access groups as shown in Table 50. The object is member of an access group if the corresponding bit is set.

Table 50 – Domain access group membership

Bit	Meaning
7	Access Group 1
6	Access Group 2
5	Access Group 3
4	Access Group 4
3	Access Group 5
2	Access Group 6
1	Access Group 7
0	Access Group 8

Access Rights

This attribute should specify the rights to access the object as shown in Table 51. The respective access is allowed if the corresponding bit is set.

Table 51 – Domain access rights membership

Bit	Name	Meaning
7	R	Right to Read the registered Password
6	W	Right to Write the registered Password
5	U	Right to Use in a PI for the registered Password
3	Rg	Right to Read for Access Groups
2	Wg	Right to Write for Access Groups
1	Ug	Right to Use in a PI for Access Groups
15	Ra	Right to Read for all Communication Partners
14	Wa	Right to Write for all Communication Partners
13	Ua	Right to Use in a PI for all Communication Partners

Local Address

This attribute is a system specific reference to the real object. It may be used internally for addressing the object. If a reference of this kind is not applied, the Local Address attribute should have the value FFFFFFFF hex.

Domain State

This attribute specifies the state of the Domain Object. The values of this attribute are:

1 <=> EXISTEN

2 <=> LOADING

3 <=> INCOMPLETE

4 <=> COMPLETE

5 <=> READY

6 <=> IN-USE

Upload State

This attribute should specify the state of the State Machine for Upload of the Domain.

0 <=> NON-EXISTENT

1 <=> UPLOADING

2 <=> UPLOADED

Counter

This attribute should specify the number of Program Invocations which currently use this Domain. The Program Invocation Management maintains this Counter, while the Domain Management only interprets the Counter. If the Counter attribute has a value greater than 0, the Domain is in use and may not be overwritten with a Download service.

Extension

This attribute specifies profile specific information.

6.3.7.2.3 Services

Generic Download Services

This service set provides for the “push” download of a domain.

Download Services

This service set provides for the “pull” download of a domain.

Upload Services

This service set provides for the “pull” upload of a domain.

6.3.7.3 Domain ASE service specifications

6.3.7.3.1 Supported services

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

- Generic Initiate Download Sequence
- Generic Download Segment
- Generic Terminate Download Sequence
- Initiate Download Sequence
- Download Segment
- Terminate Download Sequence
- Request Domain Download
- Initiate Upload Sequence
- Upload Segment
- Terminate Upload Sequence

6.3.7.3.2 Request domain uploadGeneric download services

6.3.7.3.2.1 Overview

The Generic Download Services should be used to load data from the client into the server's Domain using the “push” model.

6.3.7.3.2.2 GenericInitiateDownloadSequence

6.3.7.3.2.2.1 Service overview

The GenericInitiateDownloadSequence service should be used to begin the loading of the Domain whose Index or Domain Name is included in the service request.

6.3.7.3.2.2.2 Service primitives

The primitives and parameters of this service are shown in Table 52 and the accompanying text.

Table 52 – GenericInitiateDownloadSequence

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.2.3 GenericDownloadSegment

6.3.7.3.2.3.1 Service overview

The GenericDownloadSegment service should be used to transfer one data segment into the server's Domain. The segment is transmitted in the service request.

6.3.7.3.2.3.2 Service primitives

The primitives and parameters of this service are shown in Table 53 and the accompanying text.

Table 53 – GenericDownloadSegment

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Load Data	M	M (=)		
More Follows	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Load Data

This parameter should contain the data to be downloaded.

More Follows

This parameter should indicate whether or not any additional data remains to be transmitted.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.2.4 GenericTerminateDownloadSequence

6.3.7.3.2.4.1 Service overview

The GenericTerminateDownloadSequence service should be used to finish the loading of the Domain whose Index or Domain Name is included in the service request.

6.3.7.3.2.4.2 Service primitives

The primitives and parameters of this service are shown in Table 54 and the accompanying text.

Table 54 – GenericTerminateDownloadSequence

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Result (+)			S	S (=)
Final Result			M	M (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.				

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Final Result

This parameter should inform the client whether or not the server successfully finished the Download.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.2.4.3 Service procedure

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

6.3.7.3.3 Download services

6.3.7.3.3.1 Overview

The Download Services should be used to load data from the client into the server's Domain.

6.3.7.3.3.2 InitiateDownloadSequence

6.3.7.3.3.2.1 Service overview

The InitiateDownloadSequence service should be used to begin the loading of the Domain whose Index or Domain Name is included in the service request.

6.3.7.3.3.2.2 Service primitives

The primitives and parameters of this service are shown in Table 55 and the accompanying text.

Table 55 – InitiateDownloadSequence

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.3.3 DownloadSegment

6.3.7.3.3.3.1 Service overview

The DownloadSegment service should be used to transfer one data segment into the server's Domain. The segment is transmitted in the service response.

6.3.7.3.3.3.2 Service primitives

The primitives and parameters of this service are shown in Table 56 and the accompanying text.

Table 56 – DownloadSegment

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Result (+)			S	S (=)
Load Data	M	M (=)		
More Follows	M	M (=)		
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Load Data

This parameter should contain the data to be downloaded.

More Follows

This parameter should indicate whether or not any additional data remains to be transmitted.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.3.4 TerminateDownloadSequence

6.3.7.3.3.4.1 Service overview

The TerminateDownloadSequence service should be used to finish the loading of the Domain whose Index or Domain Name is included in the service request.

6.3.7.3.3.4.2 Service primitives

The primitives and parameters of this service are shown in Table 57 and the accompanying text.

Table 57 – TerminateDownloadSequence

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Final Result			M	M (=)
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Final Result

This parameter should inform the client whether or not the server successfully finished the Download.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.3.4.3 Service procedure

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

6.3.7.3.3.5 RequestDomainDownload

6.3.7.3.3.5.1 Service overview

The RequestDomainDownload service should be used by a server to request the client to perform a Download into the Domain whose Index or Domain Name is included in the service request.

The service response with Result(+) is not sent until the Download Sequence has been completely finished.

6.3.7.3.3.5.2 Service primitives

The primitives and parameters of this service are shown in Table 58 and the accompanying text.

Table 58 – RequestDomainDownload

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Additional Information			M	M (=)
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Additional Information

This parameter should contain additional information on the Domain. It may for example contain a file name. The correlation between Domain and file name is a local matter and is the responsibility of the application.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.3.5.3 Service procedure

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

6.3.7.3.4 Upload services

6.3.7.3.4.1 Overview

The Upload Services should be used to transmit the data from the server's Domain to the client.

6.3.7.3.4.2 InitiateUploadSequence

6.3.7.3.4.2.1 Service overview

The InitiateUploadSequence service should be used to begin the Upload of the Domain whose Index or Domain Name is included in the service request.

6.3.7.3.4.2.2 Service primitives

The primitives and parameters of this service are shown in Table 59 and the accompanying text.

Table 59 – InitiateUploadSequence

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.				

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.4.3 UploadSegment

6.3.7.3.4.3.1 Service overview

The UploadSegment service should be used to transfer one data segment of the server's Domain to the client.

6.3.7.3.4.3.2 Service primitives

The primitives and parameters of this service are shown in Table 60 and the accompanying text.

Table 60 – UploadSegment

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Result (+)			S	S (=)
Load Data	M	M (=)		
More Follows	M	M (=)		
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Load Data

This parameter should contain the data to be uploaded.

More Follows

This parameter should indicate whether or not any additional data remains to be transmitted.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.4.4 TerminateUploadSequence

6.3.7.3.4.4.1 Service overview

The TerminateUploadSequence service should be used to finish the Upload Sequence.

6.3.7.3.4.4.2 Service primitives

The primitives and parameters of this service are shown in Table 61 and the accompanying text.

Table 61 – TerminateUploadSequence

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.4.4.3 Service procedure

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

6.3.7.3.4.5 RequestDomainUpload

6.3.7.3.4.5.1 Service overview

The RequestDomainUpload service should be used by a server to request that the contents of the specified Domain be uploaded to the client.

Remark: The service response with Result(+) is not sent until the Upload Sequence has been completely finished.

6.3.7.3.4.5.2 Service primitives

The primitives and parameters of this service are shown in Table 62 and the accompanying text.

Table 62 – RequestDomainUpload

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Domain Index	M	M (=)		
Additional Information			M	M (=)
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

The argument contains the service specific parameters of the service request.

Domain Index

This parameter is the OD index of the domain object.

Additional Information

This parameter should contain additional information on the Domain. It may for example contain a file name. The correlation between Domain and file name is a local matter and is the responsibility of the application.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.7.3.4.5.3 Service procedure

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

6.3.7.4 State machine description

6.3.7.4.1 State machine for genericDownload/download

6.3.7.4.1.1 States

The states are shown in Figure 10.

EXISTENT

The Domain exists, but the content is not defined.

LOADING

GenericDownload/Download on the Domain is in progress.

INCOMPLETE

Download failed, the content is incomplete.

COMPLETE

The Domain content has been transmitted, but the Domain is not yet released for use.

READY

The Domain is released for use.

IN-USE

The Domain is being used by a Program Invocation. GenericDownload/Download is not allowed in this state.

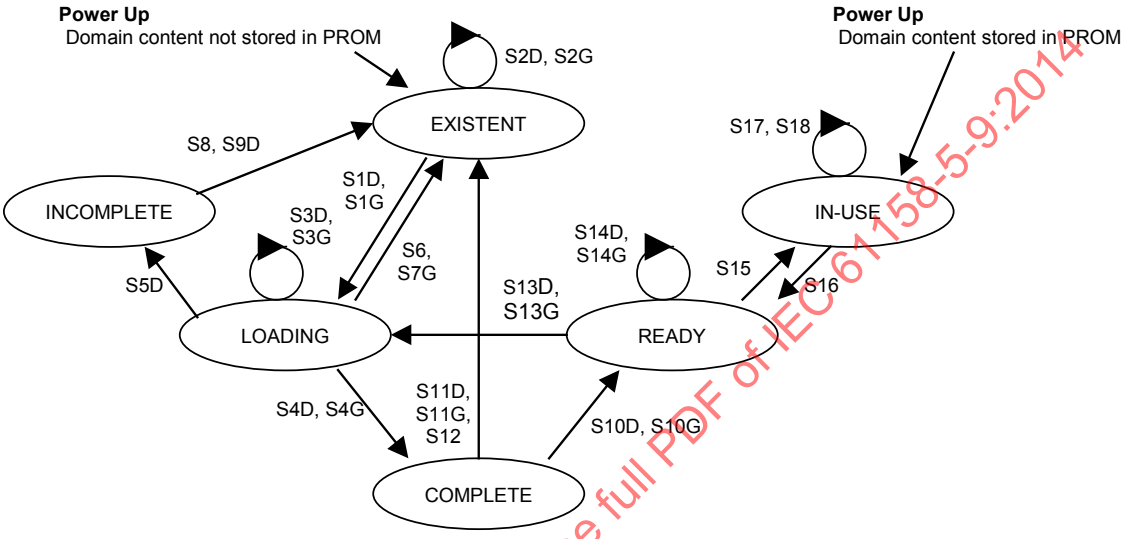


Figure 10 – Domain genericdownload/download state machine (server)

GenericDownload, Download and Upload may not operate on the same Domain at the same time.

State transition numbers with "G" indicate they are transitions for Domain GenericDownload, while transition numbers with "D" indicate they are transitions for Download.

6.3.7.4.1.2 State transitions

Table 63 specifies the server-side state transitions related to download.

Table 63 – Domain genericDownload/download state machine (server)

#	Current State	Events	Actions	Next State

S1D	EXISTENT	InitiateDownloadSequence.ind
		&& Upload state = NON-EXISTENT
			InitiateDownloadSequence.rsp(+){
			},
			DownloadSegment.req{
			}
				LOADING

S1G	EXISTENT	GenericInitiateDownloadSequence.ind
		&& Upload state = NON-EXISTENT
			GenericInitiateDownloadSequence.rsp(+
			){
			}
				LOADING

S2D	EXISTENT	InitiateDownloadSequence.ind
		&& Upload state <> NON-EXISTENT
			InitiateDownloadSequence.rsp(-){
			ErrorType := object-constraint- conflict
			}
				EXISTENT

S2G	EXISTENT	GenericInitiateDownloadSequence.ind
		&& Upload state <> NON-EXISTENT
			GenericInitiateDownloadSequence. rsp(-
			){
			ErrorType := object-constraint- conflict
			}
				EXISTENT

#	Current State	Events	Actions	Next State

S3D	LOADING	DownloadSegment.cnf(+)
		&& more follows = true
			store received data,
			DownloadSegment.req{
			}
				LOADING

S3G	LOADING	GenericDownloadSegment.ind && more follows = true && GenericDownloadSegment successfully executed	GenericDownloadSegment.rsp(+){ }	LOADING

S4D	LOADING	DownloadSegment.cnf(+)
		&& more follows = false
			store received data,
			TerminateDownloadSequence.req{
			Final Result := true
			}
				COMPLETE

S4G	LOADING	GenericDownloadSegment.ind && more follows = false && GenericDownloadSegment successfully executed	GenericDownloadSegment.rsp(+){ }	COMPLETE

S5D	LOADING	DownloadSegment.cnf(-)	TerminateDownloadSequence.req{
			Final Result := false
			}
				INCOMPLETE

S6	LOADING	Abort.ind	(no actions taken)	EXISTENT

S7G	LOADING	GenericDownloadSegment.ind
		&& more follows = true/false
		&& GenericDownloadSegment failed
			GenericDownloadSegment.rsp(-){
			ErrorType := reason of failure
			}
				EXISTENT

S8	INCOMPLETE	Abort.ind	(no actions taken)	EXISTENT
S9D	INCOMPLETE	TerminateDownloadSequence.cnf(+/-)	(no actions taken)	EXISTENT
S10D	COMPLETE	TerminateDownloadSequence.cnf(+)	(no actions taken)	READY

S10G	COMPLETE	GenericTerminateDownloadSequence.ind
		&&
		GenericTerminateDownloadSequence successfully executed
			GenericTerminateDownloadSequence.rsp(+){
			Final Result := true
			}
				READY

S11D	COMPLETE	TerminateDownloadSequence.cnf(-)	(no actions taken)	EXISTENT

S11G	COMPLETE	GenericTerminateDownloadSequence.ind
		&& GenericTerminateDownload Sequence failed
			GenericTerminateDownloadSequence.rsp(-){
			ErrorType := reason of failure
			}
				EXISTENT

S12	COMPLETE	Abort.ind	(no actions taken)	EXISTENT

S13D	READY	InitiateDownloadSequence.ind
		&& Upload state = NON-EXISTENT
			InitiateDownloadSequence.rsp(+){
			},
			DownloadSegment.req{
			}
				LOADING

S13G	READY	GenericInitiateDownloadSequence.ind && Upload state = NON-EXISTENT	GenericInitiateDownloadSequence.rsp(+){ }	LOADING

S14D	READY	InitiateDownloadSequence.ind
		&& Upload state <> NON-EXISTENT
			InitiateDownloadSequence.rsp(-){
			ErrorType := object-constraint- conflict
			}
				READY

S14G	READY	GenericInitiateDownloadSequence.ind
		&& Upload state <> NON-EXISTENT
			GenericInitiateDownloadSequence.rsp(-){
			ErrorType := object-constraint- conflict
			}
				READY

S15	READY	IncrementDomainCounter()	counter := 1	IN-USE
S16	IN-USE	DecrementDomainCounter() && counter = 1	counter := 0	READY
S17	IN-USE	IncrementDomainCounter()	counter := counter +1	IN-USE
S18	IN-USE	DecrementDomainCounter() && counter > 1	counter := counter -1	IN-USE
NOTE IncrementDomainDounter() and DecrementDomainCounter() are actions in PI state machine.				

6.3.7.4.2 State machine for upload

6.3.7.4.2.1 States

The states are shown in Figure 11.

NON-EXISTENT

The Domain exists, no Upload in progress.

UPLOADING

Upload on the Domain is in progress, Download is not allowed.

UPLOADED

The Domain content has been completely transmitted, but the TerminateUploadSequence service is still to be executed.

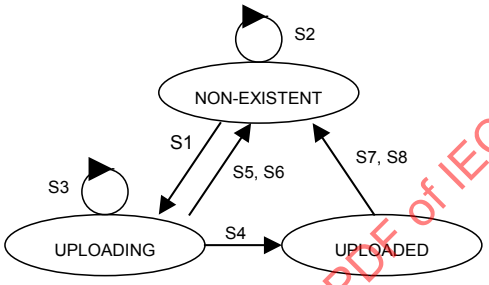


Figure 11 – Domain upload state machine (server)

Download, Upload and GenericDownload may not operate on the same Domain at the same time.

6.3.7.4.2.2 State transitions

Table 64 specifies the server-side state transitions related to upload.

Table 64 – Domain upload state machine (server)

#	Current State	Events	Actions	Next State
S1	NON-EXISTENT	InitiateUploadSequence.ind && Download state machine of this domain is in one state of {EXISTENT, READY, IN-USE}	InitiateUploadSequence.rsp(+){ }	UPLOADING

S2	NON-EXISTENT	InitiateUploadSequence.ind
		&& Download state machine of this domain is in one state of {LOADING, INCOMPLETE, COMPLETE}
			InitiateUploadSequence.rsp(-){
			ErrorType := object -constraint-conflict
			}
				NON-EXISTENT

S3	UPLOADING	UploadSegment.ind
		&& more follows = true
			UploadSegment.rsp(+){
			more follows := true
			}
				UPLOADING

S4	UPLOADING	UploadSegment.ind
		&& more follows = false
			UploadSegment.rsp(+){
			more follows := false
			}
				UPLOADED

S5	UPLOADING	TerminateUploadSequence.ind	TerminateUploadSequence.rsp(-){
			ErrorType := object-state-conflict
			}
				NON-EXISTENT

S6	UPLOADING	Abort.ind	(no actions taken)	NON-EXISTENT
S7	UPLOADED	TerminateUploadSequence.ind	TerminateUploadSequence.rsp(+){ }	NON-EXISTENT
S8	UPLOADED	Abort.ind	(no actions taken)	NON-EXISTENT

6.3.8 Program invocation ASE

6.3.8.1 Overview

Program Invocations are used to model software processes or user operations that may be started and controlled. The Program Invocation ASE defines services to invoke programs and then suspend, resume, or abort them once they have been invoked. They do not return a value in response to being started. If it is necessary to abort a program invocation once it has been started, then it should be defined as state-oriented.

6.3.8.2 Program invocation model class specifications

6.3.8.2.1 Program invocation class specification

6.3.8.2.1.1 Formal model

ASE: **PROGRAM INVOCATION ASE**
CLASS: **PROGRAM INVOCATION**
CLASS ID: **3**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Numeric Identifier
2	(m)	Attribute:	Number of Domains
3	(m)	Attribute:	Password
4	(m)	Attribute:	Access Groups
5	(m)	Attribute:	Access Rights
6	(m)	Attribute:	Deletable (TRUE,FALSE)
7	(m)	Attribute:	Reusable (TRUE,FALSE) Default : TRUE
8	(m)	Attribute:	Program Invocation State
9	(m)	Attribute:	List of Domain Index
9.1	(m)	Attribute:	Domain Index
10	(m)	Attribute:	Extension

SERVICES:

1	(m)	OpsService:	Start
2	(m)	OpsService:	Stop
3	(m)	OpsService:	Resume
4	(m)	OpsService:	Reset
5	(m)	OpsService:	Kill
6	(m)	OpsService:	Create Program Invocation
7	(m)	OpsService:	Delete Program Invocation

6.3.8.2.1.2 Attributes

Numeric Identifier

Identifies an instance of this object class.

Number of Domains

Number of the domain indices entered in this Program Invocation. The number is limited by the maximum PDU length of the Create Program Invocation service.

Password

This attribute specifies the password for the Access Rights.

Access Groups

This attribute relates the object to specific Access Groups as shown in Table 65. The object is a group member when the corresponding bit is set.

Table 65 – Program invocation access group membership

Bit	Meaning
7	Access Group 1
6	Access Group 2
5	Access Group 3
4	Access Group 4
3	Access Group 5
2	Access Group 6
1	Access Group 7
0	Access Group 8

Access Rights

This attribute specifies information about the Access Rights as shown in Table 66. The specific Access Right exists when the corresponding bit is set.

Table 66 – Program invocation access group membership

Bit	Name	Meaning
7	S	Right to Start the PI for the registered Password (Start, Resume, Reset)
6	H	Right to Stop the PI for the registered Password (Stop)
5	D	Right to Delete the PI for the registered Password (Kill, Delete Program Invocation)
3	Sg	Right to Start for Access Groups
2	Hg	Right to Stop for Access Groups
1	Dg	Right to Delete for Access Groups
15	Sa	Right to Start for all Communication Partners
14	Ha	Right to Stop for all Communication Partners
13	Da	Right to Delete for all Communication Partners

Deletable

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

Reusable

This attribute indicates if the Program Invocation transits after execution to the state IDLE or to the state UNRUNNABLE.

true <=> transits to the state IDLE

false <=> transits to the state UNRUNNABLE.

PI State

This attribute describes the state of the program.

1 ⇔ UNRUNNABLE

2 ⇔ RUNNING

3 ⇔ STARTING

4 ⇔ RESUMING

5 ⇔ IDLE

6 ⇔ STOPPED

7 ⇔ STOPPING

8 ⇔ RESETTING

List of Domain Index

This attribute specifies the indices of the domains, which are combined to this Program Invocation. The ordering of the entries of the domain indices in the Program Invocation corresponds to the ordering in the Create Program Invocation service.

The first domain in the list of the indices should contain an executable program.

Domain Index

This attribute is the OD Index for the domain.

Extension

This attribute specifies profile specific information.

6.3.8.2.1.3 Services

Create Program Invocation

This service is used to create a program invocation and relate it to one or more load regions.

Delete Program Invocation

This service is used to delete a program invocation.

Start

This service is used to start a program invocation.

Stop

This service is used to stop (suspend) a program invocation.

Resume

This service is used to resume a stopped (suspended) a program invocation.

Reset

This service is used to terminate a stopped (suspended) program invocation and prepare it to be started again.

Kill

This service is used to abort a program invocation without preparing it to be restarted.

6.3.8.3 Program invocation model service specifications

6.3.8.3.1 Create program invocation service

6.3.8.3.1.1 Service overview

With this service, domains (e.g. programs, data regions), that have been defined in the OD, are combined to a program and are defined on-line in the DP-OD. This program is addressed by a logical address or a name. The Deletable attribute of a Program Invocation, that is created by the Create Program Invocation service, is set to true. The attribute Number of Domains in the object description is created automatically by the server.

The first domain of the list of domains should contain an executable program.

The Program Invocation is only created, if the rights for the use of the domains exist. Otherwise a Result(-) is responded.

If the same Program Invocation with the same Access Rights already exists, no new object is created. In this case, the logical address (Index) of the existing object is given to the client.

6.3.8.3.1.2 Service primitives

The service parameters for this service are shown in Table 67.

Table 67 – Create program invocation service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Password	M			
Access Groups	M	M (=)		
Access Rights	M	M (=)		
Reusable	M	M (=)		
List of Domains	M	M (=)		
Domain Index	M	M (=)		
Program Invocation Name	U	U (=)		
Extension	U	U (=)		
Result (+)			S	S (=)
OD Index			M	M (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Password

This parameter states to which value the Password attribute is to be set.

Access Groups

This parameter states to which value the attribute Access Groups is to be set.

Access Rights

This parameter states to which value the attribute Access Rights is to be set.

Reusable

This parameter states to which value the Reusable attribute is to be set.

List of Domains

This parameter specifies the OD Indexes of the Domains that contain the code and data for this Program Invocation.

Domain Index

This parameter is the OD Index of the domain object.

Program Invocation Name

This parameter specifies the name of the program invocation. It is unused.

Extension

This parameter specifies the Extension attribute.

Result(+)

The Result(+) parameter should indicate that the service was completed successfully.

OD Index

OD Index of the newly created Program Invocation object.

Result(-)

The Result(-) should indicate that the service request was not completed successfully.

6.3.8.3.1.3 Service procedure

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

6.3.8.3.2 Delete program invocation service

6.3.8.3.2.1 Service overview

With the Delete Program Invocation service programs that have been defined in the DP-OD are deleted. Program Invocation Objects may only be deleted if their attribute Deletable has the value true.

6.3.8.3.2.2 Service primitives

The service parameters for this service are shown in Table 68.

Table 68 – Delete program invocation service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Program Invocation Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Program Invocation Index

This parameter specifies the OD Index of the Program Invocation object to be deleted.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.8.3.2.3 Service procedure

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

6.3.8.3.3 Start service

6.3.8.3.3.1 Service overview

A program invocation is started with this service. It runs from the beginning. The counters of the used domains are incremented. Before that all corresponding domains are checked if they are in the READY or in the IN USE state.

6.3.8.3.3.2 Service primitives

The service parameters for this service are shown in Table 69.

Table 69 – Start service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Program Invocation Index	M	M (=)		
Execution Argument	U	U (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			C	C (=)

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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Program Invocation Index

This parameter specifies the OD Index of the Program Invocation object to be started.

Execution Argument

This optional parameter of type octet string contains data that is given to the Program Invocation for the start.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

PI State

This parameter indicates the state of the Program Invocation.

6.3.8.3.4 Stop service

6.3.8.3.4.1 Service overview

A running program invocation is stopped (suspended) so that it can be resumed. It is not set to the beginning. The counters of the used domains are decremented.

6.3.8.3.4.2 Service primitives

The service parameters for this service are shown in Table 70.

Table 70 – Stop service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Program Invocation Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			C	C (=)

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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Program Invocation Index

This parameter specifies the OD Index of the Program Invocation object to be stopped.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

PI State

This parameter indicates the state of the Program Invocation.

6.3.8.3.5 Resume service

6.3.8.3.5.1 Service overview

This confirmed service is used to resume the execution of a Program Invocation that has been stopped. Execution is resumed using the context saved when the Program Invocation was stopped, as follows.

The stopped program invocation is set to the state RUNNING. It is not reset. The counters of the used domains are incremented. Before that all corresponding domains are checked if they are in the READY or in the IN USE state.

6.3.8.3.5.2 Service primitives

The service parameters for this service are shown in Table 71.

Table 71 – Resume service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Program Invocation Index	M	M (=)		
Execution Argument	U	U (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			C	C (=)
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. See 1.2.				

Argument

This parameter carries the parameters of the service invocation.

Program Invocation Index

This parameter specifies the OD Index of the Program Invocation object to be resumed.

Execution Argument

This optional parameter of type octet string contains data that is given to the Program Invocation for the resume.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

PI State

This parameter indicates the state of the Program Invocation.

6.3.8.3.6 Reset service

6.3.8.3.6.1 Service overview

This confirmed service is used to reset a stopped Program Invocation with its initial context. Its initial context is defined as the context of the Program Invocation set by its initialization procedures, as follows.

A stopped program having the attribute Reusable = true is set to the beginning. The Program Invocation transits to the state IDLE. If the attribute Reusable is equal to false the Program Invocation Object transits to the state UNRUNNABLE.

6.3.8.3.6.2 Service primitives

The service parameters for this service are shown in Table 72.

Table 72 – Reset service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Program Invocation Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			C	C (=)

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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Program Invocation Index

This parameter specifies the OD Index of the Program Invocation object to be reset.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

PI State

This parameter indicates the state of the Program Invocation.

6.3.8.3.7 Kill service

6.3.8.3.7.1 Service overview

This confirmed service is used to abort the execution of a Program Invocation so that it may not be restarted or resumed, as follows.

A Program Invocation is set to the state UNRUNNABLE. This is independent of the current state. If the state of the Program Invocation was RUNNING or STOPPING, the counters of the used domains are decremented.

6.3.8.3.7.2 Service primitives

The service parameters for this service are shown in Table 73.

Table 73 – Kill service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Program Invocation Index	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			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. See 1.2.

Argument

This parameter carries the parameters of the service invocation.

Program Invocation Index

This parameter specifies the OD Index of the Program Invocation object to be killed.

Execution Argument

This optional parameter of type octet string contains data that is given to the Program Invocation for the resume.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

6.3.8.4 Program invocation state machine

6.3.8.4.1 States

The states are shown in Figure 12.

NON-EXISTENT

The Program Invocation is after a power-up, if not otherwise defined, or after deletion, non-existent.

IDLE

A Program Invocation transits to the state IDLE by entering it in the DP-OD. After a successful Reset service and after the end of the program the Program Invocation transits also to the state IDLE. Program Invocations may be predefined. These Program Invocations transit after power-up to the state IDLE.

STARTING

Intermediate state for the preparation of the program start.

RUNNING

The program is running in this state. The domains that are defined in this Program Invocation are now in the state IN USE.

STOPPING

Intermediate state for the preparation of the stop of the program.

STOPPED

The program is stopped in this state.

RESUMING

Intermediate state for the preparation of the resumption of the program.

RESETTING

Intermediate state in which the program is set to a well-defined initial state.

UNRUNNABLE

In this state the program is stopped but it may not be started again. It may only be deleted.

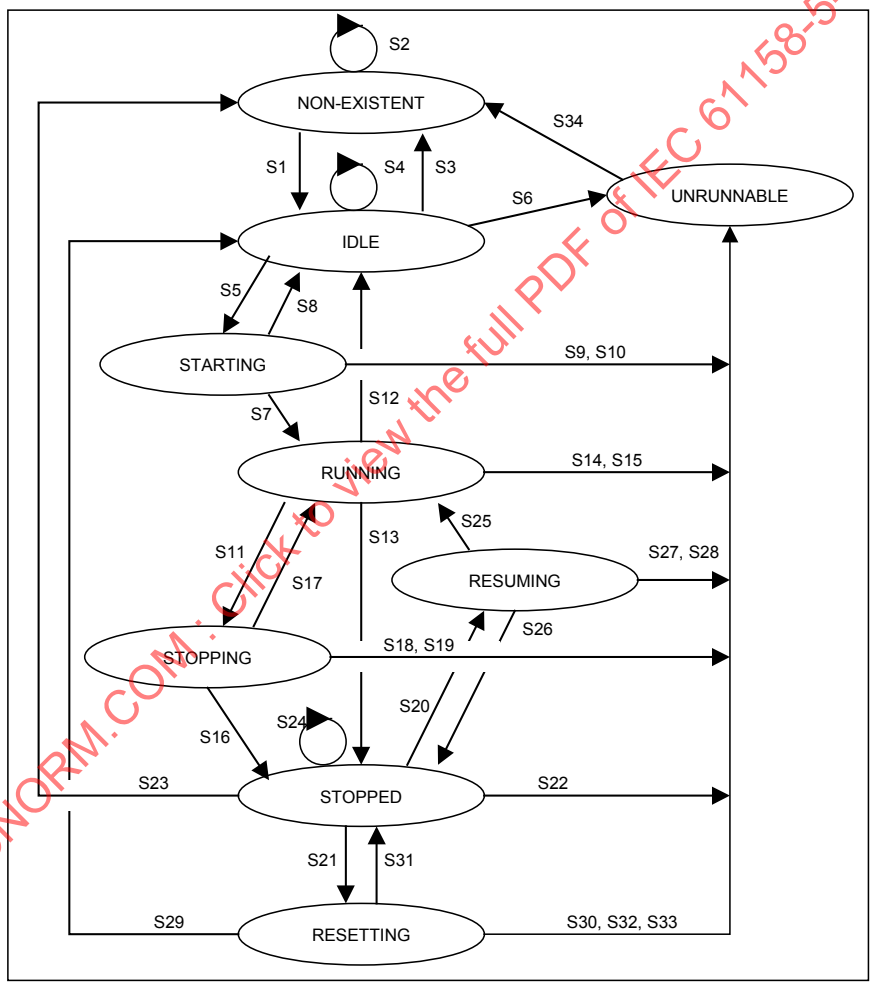


Figure 12 – State diagram

6.3.8.4.2 State transitions

Table 74 specifies the state transitions related to program invocation.

Table 74 – Program invocation state machine

#	Current State	Events	Actions	Next State
S1	NON-EXISTENT	CreateProgramInvocation.ind && Domains available && Access to Domains allowed	CreateProgramInvocation.rsp(+){ } NON-EXISTENT	IDLE

S2	NON-EXISTENT	CreateProgramInvocation.ind
		&& (Domains not available
		|| Access to Domains not allowed)
			CreateProgramInvocation.rsp(-){
			ErrorType := object-constraint-conflict
			}
				NON-EXISTENT

S3	IDLE	DeleteProgramInvocation.ind && Deletable = true	DeleteProgramInvocation.rsp(+){ }	NON-EXISTENT

S4	IDLE	DeleteProgramInvocation.ind
		&& Deletable = false
			DeleteProgramInvocation.rsp(-){
			ErrorType := object-constraint-conflict
			}
				IDLE

S5	IDLE	Start.ind	(no actions taken)	STARTING
S6	IDLE	Kill.ind	Kill.rsp(+){ }	UNRUNNABLE
S7	STARTING	Start successfully executed && Domains in State READY or IN-USE	IncrementDomainCounter(), Start.rsp(+){ } (note)	RUNNING

S8	STARTING	Start failed, non-destructive
		(e.g. Domains not in State READY or IN-USE)
			Start.rsp(-){
			ErrorType := object-constraint-conflict
			}
				IDLE

S9	STARTING	Start failed, destructive	Start.rsp(-){
			ErrorType := object-access-denied
			}
				UNRUNNABLE

S10	STARTING	Kill.ind	Kill.rsp(+){ }	UNRUNNABLE
S11	RUNNING	Stop.ind	(no actions taken)	STOPPING
S12	RUNNING	End of Program && Reusable = true	DecrementDomainCounter()	IDLE
S13	RUNNING	Program stop	DecrementDomainCounter()	STOPPED
S14	RUNNING	End of Program && Reusable = false	DecrementDomainCounter()	UNRUNNABLE

S15	RUNNING	Kill.ind	DecrementDomainCounter(),
			Kill.rsp(+){
			}
				UNRUNNABLE

S16	STOPPING	Stop successfully executed	DecrementDomainCounter(),
			Stop.rsp(+){
			}
				STOPPED

S17	STOPPING	Stop failed, non-destructive	Stop.rsp(-){
			ErrorType := object-constraint-conflict
			}
				RUNNING

S18	STOPPING	Stop failed, destructive	DecrementDomainCounter(),
			Stop.rsp(-){
			ErrorType := object-access-denied
			}
				UNRUNNABLE

S19	STOPPING	Kill.ind	DecrementDomainCounter(),
			Kill.rsp(+){
			}
				UNRUNNABLE

S20	STOPPED	Resume.ind	(no actions taken)	RESUMING
S21	STOPPED	Reset.ind	(no actions taken)	RESETTING
S22	STOPPED	Kill.ind	Kill.rsp(+){ }	UNRUNNABLE
S23	STOPPED	DeleteProgramInvocation.ind && Deletable = true	DeleteProgramInvocation.rsp(+){ }	NON-EXISTENT

S24	STOPPED	DeleteProgramInvocation.ind
		&& Deletable = false
			DeleteProgramInvocation.rsp(-){
			ErrorType := object-constraint-conflict
			}
				STOPPED

#	Current State	Events	Actions	Next State

S25	RESUMING	Resume successfully executed
		&& Domains in State READY or IN-USE
			IncrementDomainCounter(),
			Resume.rsp(+){
			}
				RUNNING

S26	RESUMING	Resume failed, non-destructive
		(e.g. Domains not in State READY or IN-USE)
			Resume.rsp(-){
			ErrorType := object-constraint-conflict
			}
				STOPPED

S27	RESUMING	Resume failed, destructive	DecrementDomainCounter(),
			Stop.rsp(-){
			ErrorType := object-access-denied
			}
				UNRUNNABLE

S28	RESUMING	Kill.ind	Kill.rsp(+){ }	UNRUNNABLE
S29	RESETTING	Reset successfully executed && Reusable = true	Reset.rsp(+){ }	IDLE
S30	RESETTING	Reset successfully executed && Reusable = false	Reset.rsp(+){ }	UNRUNNABLE

S31	RESETTING	Reset failed, non-destructive	Reset.rsp(-){
			ErrorType := object-constraint-conflict
			}
				STOPPED

S32	RESETTING	Reset failed, destructive	Reset.rsp(-){
			ErrorType := object-access-denied
			}
				UNRUNNABLE

S33	RESETTING	Kill.ind	Kill.rsp(+){ }	UNRUNNABLE
S34	UNRUNNABLE	DeleteProgramInvocation.ind	DeleteProgramInvocation.rsp(+){ }	NON-EXISTENT
NOTE IncrementDomainCounter() and DecrementDomainCounter() are actions for Download state machine.				

6.4 ARs

6.4.1 Overview

The following AREPs are used:

- Queued User-triggered Uni-directional (QUU) AR Endpoint
- Queued User-triggered Bi-directional Connection-Oriented (QUB-Co) AR Endpoint
- Buffered Network-Scheduled Uni-directional (BNU) AR Endpoint

6.4.2 Queued user-triggered unidirectional AREP class specification

6.4.2.1 Formal model

ASE: **AR ASE**
CLASS: **QueuedUser-triggeredUnidirectionalAREP**
CLASS ID:
PARENT CLASS: **AR Endpoint**
ATTRIBUTES:
1 (m) Attribute: Role (Client, Server, Peer)
2 (m) Attribute: RemoteAddressConfigurationType (Free, Linked)
3 (m) Attribute: State
SERVICES:
1 (m) OpsService: Unconfirmed Send

6.4.2.2 Attributes

Role

This attribute specifies possible role of this end point. The possible values are Source and Sink.

Source AREPs of this type convey their data by issuing a Data Transfer Unconfirmed PDU.

Sink AREPs of this type receive data transmitted by a Source AREP.

RemoteAddressConfigurationType

This attribute specifies how the remote address of the AREP, that is used to establish an AR, is specified.

Free The value of "Free" indicates that the destination of the FAS-PDUs from the Source AREP is provided during the DTU service invocations.

Linked The value of "Linked" indicates that the destination of the FAS-PDUs is configured with the RemoteDlsapAddress attribute of the QuuCl class defined later.

Initiator

This conditional attribute is present if the Role attribute has the value of either Client or Peer. The value of True means that this AREP can issue but cannot receive an Associate request PDU. The value of False means that this AREP can receive but cannot issue an Associate request PDU.

State

This attribute indicates the current state of the Application Relationship Protocol Machine (ARPM) that is defined in detail in the later section. The possible values for this attribute are specified there.

6.4.2.3 Services

Unconfirmed Send

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

6.4.3 Queued user-triggered bidirectional AREP class specification

6.4.3.1 Formal model

ASE: **AR ASE**
CLASS: **QueuedUser-triggeredBidirectionalAREP**
CLASS ID:
PARENT CLASS: **AR Endpoint**
ATTRIBUTES:

1	(m)	Attribute:	Role (Client, Server, Peer)
2	(m)	Attribute:	RemoteAddressConfigurationType (Free, Linked)
3	(m)	Attribute:	Role = Peer || Client
3.1	(m)	Attribute:	Initiator (True, False)
4	(m)	Attribute:	State

SERVICES:

1	(m)	OpsService:	Confirmed Send

6.4.3.2 Attributes

Role

This attribute specifies possible role of this end point. The possible values are Client, Server, and Peer.

Peer AREPs of this type may perform as either Clients or Servers.

Client AREPs of this type issue Request-FAS-PDUs to Servers and receive their corresponding Response-FAS-PDUs.

Server AREPs of this type receive Request-FAS-PDUs from Clients and issue corresponding Response-FAS-PDUs to them.

RemoteAddressConfigurationType

This attribute specifies how the remote address of the AREP, that is used to establish an AR, is specified.

Free The value of “Free” indicates that the destination address is provided with the Associate service request.

Linked The value of “Linked” indicates that the destination address of the Associate service request is specified by the RemoteDlcepAddress attribute of the QubCo class defined later.

Initiator

This conditional attribute is present if the Role attribute has the value of either Client or Peer. The value of True means that this AREP can issue but cannot receive an Associate request PDU. The value of False means that this AREP can receive but cannot issue an Associate request PDU.

State

This attribute indicates the current state of the Application Relationship Protocol Machine (ARPM) that is defined in detail in the later section. The possible values for this attribute are specified there.

6.4.3.3 Services

Confirmed Send

This service is used to send a confirmed service on the specified AR.

6.4.4 Buffered network-scheduled unidirectional AREP class specification

6.4.4.1 Formal model

ASE:			**AR ASE**
CLASS:			**QueuedUser-triggeredUnidirectionalAREP**
CLASS ID:			
PARENT CLASS:			**AR Endpoint**
ATTRIBUTES:			
1	(m)	Attribute:	Role (Client, Server, Peer)
2	(m)	Attribute:	DuplicatePduDetectionSupported (True, False)
3	(m)	Attribute:	State

SERVICES:

1	(m)	OpsService:	Unconfirmed Send
2	(o)	OpsService:	Compel
3	(o)	OpsService:	Get-Buffered-Message
4	(o)	OpsService:	AR-Status

6.4.4.2 Attributes

Role

This attribute specifies possible role of this end point. The possible values are Publisher and Subscriber.

Publisher AREPs of this type publish their data issuing unconfirmed service request-PDUs.

Subscriber AREPs of this type receive data published in unconfirmed service request-PDUs.

DuplicatePduDetectionSupported

This attribute specifies when True that the AREP is capable of relaying status information to the AR user indicating whether or not the DLL has delivered a duplicate DL-PDU to the AREP.

State

This attribute indicates the current state of the Application Relationship Protocol Machine (ARPM) that is defined in detail in the later section. The possible values for this attribute are specified there.

6.4.4.3 Services

Unconfirmed Send

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

Compel

This optional service is used to evoke acyclic communication out on a BNU AREP.

Get-Buffered-Message

This optional service is used to retrieve the current contents of the associated buffer.

AR-Status

This optional service is used to report status of the AR.

6.5 Summary of classes

This subclause contains a summary of the defined Classes. Table 75 provides a summary of the classes.

Table 75 – Class summary

ASE	Class	Class ID
VFD	VFD	—
Data type	Fixed Length & String Data type	5
	Structure Data type	6
Object Dictionary	OD Description	—
	Object Dictionary	—
Context Management	VCR List	—
Application Relationship	AREP	32
	QUB-Co	34
	QUU	36
	BNU	38
Variable	Simple Variable	7
	Array Variable	8
	Record Variable	9
	Variable List	10
	Data type	
	Data type Structure	
Event	Event	4
Domain	Domain	2
Program Invocation	Program Invocation	3

6.6 Permitted services by AREP role

Table 76 defines the valid combinations of services and AREP roles (which service APDUs and AREP with the specified role can send or receive). The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 76 – Services by AREP role

	Unc	Cnf	Client	Server	Push Publ	Push Subsc	Pull Publ Mgr	Pull Publ	Pull Subsc	Report Src	Report Sink
Services			req rcv	req rcv	req rcv	req rcv	req rcv	req rcv	req rcv	req rcv	req rcv
VFD ASE											
Status		X	X	X							
Unsolicited Status	X		X	X							
Identify		X	X	X							

	Unc	Cnf	Client		Server		Push Publ	Push Subsc	Pull Publ Mgr		Pull Publ	Pull Subsc	Report Src	Report Sink
Services			req	rcv	req	rcv	req rcv	req rcv	req	rcv	req rcv	req rcv	req rcv	req rcv
Object Dictionary ASE														
Get OD		X	X			X								
Initiate Put OD		X	X			X								
Put OD		X	X			X								
Terminate Put OD		X	X			X								
Context Mgt ASE														
Initiate		X	X			X								
Abort		X	X			X								
Reject		X	X			X								
AR ASE														
AR-Confirmed Send			X			X								
AR-Unconfirmed Send			X	X	X	X	X	X						
AR-Associate				X		X								
AR-Abort			X	X	X	X	X	X	X				X	X
AR Compel			X		X			X						
AR-Get Buffered Msg			X		X			X						
AR-Status			X		X									
Variable ASE														
Read		X	X			X			X	X	X			
Write		X	X			X								
Information Report	X						X	X					X	X
Define Variable List		X	X			X								
Delete Variable List		X	X			X								
Event ASE														
Event Notification	X						X	X					X	X
Ack Event		X	X			X								
Alter Event Cond Mon		X	X			X								
Domain ASE														
GenInitDwnldSeq		X	X			X								
GenDownloadSeg		X	X			X								
GenTermDwnldSeq		X	X			X								
InitiateDwnldSeq		X	X			X								
DwnldSeg		X	X			X								
TermDwnldSeq		X	X			X								
ReqDomainDwnld		X	X			X								
InitiateUploadSeq		X	X			X								
UploadSeg		X	X			X								
TermUploadSeq		X	X			X								
ReqtDomainUpload		X	X			X								
Program Inv ASE														
Start		X	X			X								
Stop		X	X			X								
Resume		X	X			X								
Reset		X	X			X								
Kill		X	X			X								
Create Program Inv		X	X			X								
Delete Program Inv		X	X			X								

Bibliography

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

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

IEC 61158-5-5, *Industrial communication networks – Fieldbus specifications – Part 5-5: Application layer service specification – Type 5 elements*

IEC 61158-6-9, *Industrial communication networks – Fieldbus specifications – Part 6-9: Application layer protocol specification – Type 9 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 – Part 3: Naming and addressing*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-9: Définition des services de la couche application – Eléments de type 9

AVANT-PROPOS

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

L'attention est attirée sur le fait que l'utilisation du type de protocole associé est restreinte par les détenteurs des droits de propriété intellectuelle. En tout état de cause, l'engagement de renonciation partielle aux droits de propriété intellectuelle pris par les détenteurs de ces droits autorise l'utilisation d'un type de protocole de couche avec les autres protocoles de couche du même type, ou dans des combinaisons avec d'autres types autorisées explicitement par les détenteurs des droits de propriété intellectuelle pour ce type.

NOTE Les combinaisons de types de protocoles sont spécifiées dans la CEI 61784–1 et la CEI 61784–2.

La Norme internationale CEI 61158-5-9 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette deuxième édition annule et remplace la première édition parue en 2007. Cette édition constitue une révision technique. La modification principale par rapport à l'édition précédente est la suivante:

- Correction des services d'événements du diagramme d'états de téléchargement descendant

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65C/763/FDIS	65C/773/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

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

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

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

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

INTRODUCTION

La présente partie de la CEI 61158 s'inscrit dans une série créée pour faciliter l'interconnexion des composants de systèmes d'automatisation. Elle est relative aux autres normes de l'ensemble défini par le modèle de référence de bus de terrain "à trois couches" décrit dans la CEI 61158-1.

Le protocole d'application fournit le service d'application au moyen des services disponibles au niveau de la couche liaison de données ou de la couche immédiatement inférieure. La présente norme définit les caractéristiques du service d'application que les applications de bus de terrain et/ou la gestion du système peuvent exploiter.

Dans l'ensemble des normes relatives aux bus de terrain, le terme "service" fait référence à la capacité abstraite fournie par une couche du Modèle de Référence de Base OSI à la couche immédiatement supérieure. Par conséquent, le service de couche Application défini dans la présente norme est un service architectural conceptuel, indépendant des divisions administratives et des divisions de mise en œuvre.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-9: Définition des services de la couche application – Eléments de type 9

1 Domaine d'application

1.1 Généralités

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

La présente norme fournit des éléments communs pour les communications de messagerie de base à temps critique et à temps non critique entre des programmes d'application dans un environnement d'automation et le matériel spécifique aux bus de terrain de Type 9. Le terme "à temps critique" sert à représenter la présence d'une fenêtre temporelle dans les limites de laquelle une ou plusieurs actions spécifiées sont exigées d'être parachevées avec un certain niveau défini de certitude. Le manquement à parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent ces actions, avec le risque concomitant pour l'équipement, les installations et éventuellement pour la vie humaine.

La présente norme définit de manière abstraite le service visible de l'extérieur fourni par les différents types de couche Application de bus de terrain, en termes

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

b) des actions et des événements de primitives du service;

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

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

La présente norme vise à définir les services fournis à

1) l'utilisateur de FAL à la frontière entre l'utilisateur et la Couche Application du Modèle de Référence de Bus de Terrain, et

2) la Gestion des Systèmes à la frontière entre la Couche Application et la Gestion des Systèmes du Modèle de Référence de Bus de Terrain.

La présente norme spécifie la structure et les services de la couche Application de bus de terrain de la CEI, en conformité avec le Modèle de Référence de Base OSI (ISO/CEI 7498-1) et avec la Structure de la Couche Application OSI (ISO/CEI 9545).

Les services et les protocoles de la couche FAL sont fournis par les entités d'application (*Application-entities*) (AE) de FAL contenues dans des processus d'application. L'AE de la couche FAL se compose d'un ensemble d'Eléments de Service d'Application (ASE) orientés objet et d'une Entité de Gestion de Couche *(Layer Management Entity)* (LME) qui gère l'AE. Les ASE fournissent des services de communication qui fonctionnent sur un ensemble de classes d'objets de processus d'application *(Application Process Object)* (APO) associées. L'un des ASE de la couche FAL est un ASE de gestion qui fournit un ensemble de services commun pour la gestion des instances des classes de FAL.

Bien que ces services spécifient, du point de vue des applications, la manière dont la demande et les réponses sont émises et délivrées, ils ne spécifient pas ce que les applications qui demandent et qui répondent doivent en faire. A savoir, les aspects comportementaux des applications ne sont pas spécifiés; seule une définition des demandes et réponses qu'elles peuvent envoyer/recevoir est spécifiée. Cela permet une plus grande flexibilité aux utilisateurs de la FAL pour normaliser un tel comportement d'objet. En plus de ces services, certains services d'appui sont également définis dans la présente norme pour fournir un accès à la FAL afin de maîtriser certains aspects de son fonctionnement.

1.2 Spécifications

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

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles de communications industrielles préexistants. C'est ce dernier objectif qui donne naissance à la diversité des services normalisés comme les divers types de la CEI 61158.

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

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

b) la corrélation de primitives appariées "request-confirm" (c'est-à-dire demande et confirmation) ou "indication-response" (indication et réponse).

1.3 Conformité

La présente norme ne définit pas de mises en œuvre ni de produits particuliers, pas plus qu'elle ne limite les mises en œuvre des entités de couche Application dans les systèmes d'automatisation industriels.

Il n'y a aucune conformité de l'équipement à la présente norme de définition des services de couche application. A la place, la conformité est obtenue par une mise en œuvre de protocoles conformes de couche application qui satisfont aux services de couche application de Type 9 tels que définis dans la présente norme.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

NOTE Toutes les parties de la série CEI 61158, ainsi que la CEI 61784-1 et la CEI 61784-2 font l'objet d'une maintenance simultanée. Les références croisées à ces documents dans le texte se rapportent par conséquent aux éditions datées dans la présente liste de références normatives.

CEI 61158-1:2014, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Présentation et lignes directrices des séries CEI 61158 et CEI 61784*

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange* (disponible en anglais seulement)

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

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

ISO/IEC 8824-1, *Information Technology – Abstract Syntax Notation One (ASN1): Specification of basic notation* (disponible en anglais seulement)

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

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

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic* (disponible en anglais seulement)

3 Termes, définitions, symboles, abréviations et conventions

Pour les besoins du présent document, les termes, définitions, symboles, abréviations et conventions suivants comme définis dans ces publications s'appliquent.

3.1 Termes de l'ISO/CEI 7498-1

a) entité d'application
b) processus d'application
c) unité de données de protocole d'application
d) élément de service d'application
e) invocation d'entités d'application
f) invocation de processus d'application
g) transaction d'applications
h) système ouvert réel
i) syntaxe de transfert

3.2 Termes de l'ISO/CEI 8822

a) syntaxe abstraite
b) contexte de présentation

3.3 Termes de l'ISO/CEI 9545

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

i) élément de service de contrôle d'application

3.4 Termes de l'ISO/CEI 8824-1

a) identificateur d'objet

b) type

3.5 Termes de la CEI 61158-1

Pour les besoins du présent document, les termes suivants, donnés dans la CEI 61158-1, s'appliquent.

3.5.1
application
fonction ou structure de données pour laquelle des données sont consommées ou produites

3.5.2
interopérabilité de couche application
capacité des entités d'application d'accomplir des opérations coordonnées et coopératives en utilisant les services de la FAL

3.5.3
objet d'application
classe d'objets qui gère et assure l'échange de messages sur le réseau et dans l'appareil réseau, lors de l'exécution

Note 1 à l'article: Plusieurs types de classes d'objets d'application peuvent être définis.

3.5.4
processus d'application
partie d'une application distribuée sur un réseau, située sur un appareil et associée à une adresse non ambiguë

3.5.5
identificateur de processus d'application
composant qui distingue de multiples processus d'application utilisés dans un appareil

3.5.6
objet de processus d'application
composant d'un processus d'application, identifiable et accessible par le biais d'une relation entre applications FAL

Note 1 à l'article: Les définitions d'objets de processus d'application se composent d'un ensemble de valeurs destinées aux attributs de leur classe (voir la définition "Classe d'Objets de Processus d'Application"). Les définitions d'objets de processus d'application sont accessibles à distance à l'aide des services de l'ASE "FAL Object Management" (Gestion d'objets de la FAL). Les services de Gestion d'Objets de la FAL peuvent être utilisés pour charger ou mettre à jour des définitions d'objets, pour lire des définitions d'objets, ainsi que pour créer et supprimer de manière dynamique des objets d'application et leurs définitions correspondantes.

3.5.7
classe d'objets de processus d'application
classe d'objets de processus d'application définis par un ensemble de services et d'attributs accessibles par le biais du réseau

3.5.8
relation entre applications
association coopérative entre deux ou plusieurs invocations d'entités d'application (application-entity-invocation) à des fins d'échange d'informations et de coordination de leur fonctionnement conjoint

Note 1 à l'article: Cette relation est activée soit par l'échange d'unités de données de protocole d'application, soit à la suite d'activités de préconfiguration.

3.5.9
élément de service d'application de relation entre applications
application-service-element (élément de service d'application) qui fournit le moyen exclusif d'établir et de faire cesser toutes les relations d'applications

3.5.10
point d'extrémité de relation entre applications
contexte et comportement d'une relation entre applications, vus et maintenus par un des processus d'application impliqués dans la relation entre applications

Note 1 à l'article: Chaque processus d'application impliqué dans la relation entre applications maintient son propre point d'extrémité de relation entre applications.

3.5.11
attribut
description d'une caractéristique ou d'une fonctionnalité, visible de l'extérieur, d'un objet

Note 1 à l'article: Les attributs d'un objet contiennent des informations sur les portions variables d'un objet. Typiquement, ils fournissent des informations relatives au statut ou régissent le fonctionnement d'un objet. Les attributs peuvent également influer sur le comportement d'un objet. Les attributs se répartissent en deux catégories: les attributs de classe et les attributs d'instance.

3.5.12
comportement
indication de la manière dont l'objet réagit à des événements particuliers

Note 1 à l'article: Sa description inclut la relation entre les valeurs des attributs et les services.

3.5.13
classe
ensemble d'objets représentant le même type de composant du système

Note 1 à l'article: Une classe est une généralisation de l'objet, un modèle qui permet de définir des variables et des méthodes. Tous les objets d'une classe possèdent une forme et un comportement identiques, mais leurs attributs contiennent généralement des données différentes.

3.5.14
attributs de classe
attribut commun à tous les objets d'une même classe

3.5.15
code de classe
identificateur non ambigu attribué à chaque classe d'objets

3.5.16
service propre à une classe
service défini par une classe d'objets particulière afin de remplir une fonction exigée qui n'est assurée par aucun service commun

Note 1 à l'article: Tout objet propre à une classe est réservé à l'usage exclusif de la classe d'objets qui le définit.

3.5.17
client
(a) objet qui utilise les services d'un autre objet (serveur) pour accomplir une tâche

(b) initiateur d'un message auquel un serveur réagit; par exemple, rôle d'un point d'extrémité de relation entre applications, dans lequel ce dernier envoie des unités APDU de demande de service confirmé à un point d'extrémité de relation entre applications jouant le rôle de serveur

3.5.18
chemin de transport
flux unidirectionnel d'unités APDU à travers une relation entre applications

3.5.19
cyclique
terme utilisé pour décrire les événements qui se reproduisent de manière régulière et répétitive

3.5.20
AR dédiée
AR directement utilisée par l'utilisateur de FAL

Note 1 à l'article: Dans les AR dédiées, seuls l'en-tête de FAL et les données de l'utilisateur sont transférés.

3.5.21
appareil
connexion matérielle physique à la liaison

Note 1 à l'article: Un appareil peut contenir plusieurs nœuds.

3.5.22
profil d'appareil
ensemble d'informations et de fonctionnalités dépendant de l'appareil qui garantit la cohérence entre les appareils similaires appartenant au même type d'appareil

3.5.23
AR dynamique
AR qui doit être mise dans un état établi par le biais des procédures d'établissement d'AR

3.5.24
point d'extrémité
l'une des entités en communication impliquées dans une connexion

3.5.25
erreur
divergence entre une valeur ou condition calculée, observée ou mesurée et la valeur ou condition spécifiée ou théoriquement correcte

3.5.26
classe d'erreurs
groupement général de définitions d'erreurs

Note 1 à l'article: Les codes d'erreur associés à des erreurs particulières sont définis dans une classe d'erreurs.

3.5.27
code d'erreur
identification d'un type spécifique d'erreur dans une classe d'erreurs

3.5.28
sous-réseau FAL
réseaux composés d'un ou de plusieurs segments de liaison de données

Note 1 à l'article: Ces réseaux peuvent contenir des ponts, mais pas de routeurs. Les sous-réseaux FAL sont identifiés par un sous-ensemble de l'adresse de réseau.

3.5.29
appareil logique
classe FAL particulière qui extrait un composant logiciel ou un composant de microprogramme sous la forme d'un dispositif intégré et autonome au sein d'un appareil d'automatisation

3.5.30
informations de gestion
informations accessibles par le biais du réseau qui facilitent la gestion de l'exploitation du système de bus de terrain, y compris la couche application

Note 1 à l'article: La gestion inclut des fonctions telles que le contrôle, la surveillance et le diagnostic.

3.5.31
réseau
série de nœuds reliés par un certain type de support de communication

Note 1 à l'article: Les chemins de connexion entre des paires de nœuds peuvent inclure des répéteurs, des routeurs et des passerelles.

3.5.32
homologue
rôle d'un point d'extrémité d'AR dans lequel il est capable d'agir tant comme client que comme serveur

3.5.33
point d'extrémité d'AR prédéfini
point d'extrémité d'AR qui est défini localement dans un appareil sans utilisation du service de création

Note 1 à l'article: Les AR prédéfinies qui ne sont pas préétablies sont établies avant d'être utilisées.

3.5.34
point d'extrémité d'AR préétabli
point d'extrémité d'AR qui est mis dans un état établi pendant la configuration des AE qui contrôlent ses points d'extrémité

3.5.35
éditeur
rôle d'un point d'extrémité d'AR dans lequel il transmet des unités APDU sur le bus de terrain afin qu'elles soient consommées par un ou plusieurs abonnés

Note 1 à l'article: L'éditeur peut ne pas connaître l'identité des abonnés ou leur nombre; il peut publier ses unités APDU au moyen d'une AR dédiée. La présente norme définit deux types d'éditeurs: les éditeurs par extraction (*Pull Publishers*) et les éditeurs par émission (*Push Publishers*); chacun faisant l'objet d'une définition distincte.

3.5.36
serveur
a) rôle d'un AREP dans lequel il renvoie une unité APDU de réponse de service confirmé au client à l'origine de la demande

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

3.5.37
service
opération ou fonction qu'un objet et/ou une classe d'objets exécute(nt) à la demande d'un autre objet et/ou d'une autre classe d'objets

Note 1 à l'article: Un ensemble de services communs est défini et des dispositions permettant la définition de services propres à un objet sont fournies. Les services propres à un objet sont des services définis par une classe d'objets particulière afin de remplir une fonction exigée qui n'est assurée par aucun service commun.

3.5.38
abonné
rôle d'un AREP dans lequel il reçoit les unités APDU produites par un éditeur

Note 1 à l'article: La présente norme définit deux types d'abonnés: les abonnés par extraction (Pull subscriber) et les abonnés par émission (Push subscriber); chacun faisant l'objet d'une définition distincte.

3.6 Définitions spécifiques à la couche Application de bus de terrain de Type 9

Il n'y a pas de termes supplémentaires définis.

3.7 Abréviations et symboles

AE	Application Entity (Entité d'application)
AL	Application Layer (Couche Application)
ALME	Application Layer Management Entity (Entité de Gestion de Couche Application)
ALP	Application Layer Protocol (Protocole de Couche Application)
APO	Objet d'application *(Application Object)*
AP	Application Process (Processus d'Application)
APDU	Application Protocol Data Unit (Unité de Donnée de Protocole Application)
API	Application Process Identifier (Identificateur de Processus d'Application)
AR	Application Relationship (Relation entre Applications)
AREP	Application Relationship End Point (Point d'extrémité de Relation entre Applications)
ASCII	American Standard Code for Information Interchange (Code Standard Américain pour l'Echange d'Informations)
ASE	Application Service Element (Elément de Service Application)
Cnf	Confirmation
DL-	Préfixe indiquant la couche liaison de données
DLC	Data Link Connection (Connexion de Liaison de Données)
DLCEP	Data-Link Connection End Point (Point d'extrémité de Connexion de Couche Liaison de Données)
DLL	Data Link Layer (Couche Liaison de Données)
DLM	Data Link-management (Gestion de Liaison de Données)
DLSAP	Data link Service Access Point (Point d'Accès au Service Liaison de Données)
DLSDU	DL-service-data-unit (Unité de Données de Service de DL)
FAL	Fieldbus Application Layer (Couche Application de Bus de Terrain)
ID	Identificateur
CEI	Commission Electrotechnique Internationale
Ind	Indication
LME	Layer Management Entity (Entité de Gestion de Couche)
OSI	Open Systems Interconnect (Interconnexion de Systèmes Ouverts)
QoS	Quality of Service (Qualité de service)
Req	Request (Demande)
Rsp	Response (Réponse)
SAP	Service Access Point (Point d'Accès au Service)
SDU	Service Data Unit (Unité de Données de Service)
SMIB	System Management Information Base (Base d'Informations de Gestion de Système)
SMK	System Management Kernel (Noyau de Gestion Système)
VFD	*Virtual Field Device* (Appareil de champ virtuel)

3.8 Conventions

3.8.1 Vue d'ensemble

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

La spécification de classe définit les attributs de la classe. Les attributs sont accessibles à partir d'instances de la classe en utilisant les services d'ASE du Dictionnaire d'Objets spécifiés en 6.3.2 de la présente norme. La spécification de services définit les services qui sont fournis par l'ASE.

3.8.2 Conventions relatives aux définitions de classe

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

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

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

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

(3) L'article "CLASS ID:" est un numéro qui identifie la classe spécifiée. Ce numéro est unique au sein de l'élément ASE de la couche FAL qui fournira les services pour cette classe. Lorsqu'il est qualifié par l'identité de son ASE FAL, il identifie sans ambiguïté la classe relevant du domaine d'application de la FAL. La valeur "NULL" indique que la classe ne peut pas être instanciée. Les "Class ID" (identificateurs de classe) entre 1 et 255 sont réservés par la présente norme pour identifier des classes normalisées. Ils ont été attribués pour conserver la compatibilité avec des normes nationales existantes. Les "CLASS ID" entre 256 et 2 048 sont alloués pour identifier les classes définies par l'utilisateur.

(4) L'article "CLASSE PARENTE:" est le nom de la classe parente pour la classe spécifiée. Tous les attributs définis pour la classe parente et hérités par celle-ci sont hérités pour la classe définie, et ils n'ont donc pas à être redéfinis dans le modèle pour cette classe.

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

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

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

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

c) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par période. L'imbrication est utilisée pour spécifier

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

ii) des attributs conditionnés à un énoncé de contrainte (5). Les attributs peuvent être obligatoires (5.1) ou facultatifs (5.2) si la contrainte est vraie. Tous les attributs facultatifs n'exigent pas des énoncés de contraintes comme le fait l'attribut défini en (5.2).

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

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

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

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

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

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

3.8.3 Conventions pour les définitions des services

3.8.3.1 Généralités

La présente norme emploie les conventions de description énoncées dans l'ISO/CEI 10731.

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

3.8.3.2 Paramètres de service

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

NOTE 1 Voir la note en 3.8.3.3 relative à la non inclusion des paramètres de service appropriés à une spécification de protocoles ou d'interfaces de programmation ou de mises en œuvre, mais pas à une définition de services abstraits.

La présente norme utilise un format de tableau pour décrire les paramètres de composants des primitives du service. Les paramètres qui s'appliquent à chaque groupe de primitives de service sont consignés en tableaux dans le rappel de la présente norme. Chaque tableau comporte jusqu'à six colonnes: une colonne pour le nom du paramètre de service, et une colonne pour chacune des primitives et sens de transfert des paramètres utilisés par le service. Les six colonnes possibles sont:

1) le nom du paramètre;
2) les paramètres d'entrée de la primitive "request" (demande);
3) les paramètres de sortie de la primitive "request" (demande);

NOTE 2 Il s'agit d'une capacité rarement utilisée. Sauf spécification contraire, les paramètres de la primitive "request" (demande) sont des paramètres d'entrée.

4) les paramètres de sortie de la primitive "indication";
5) les paramètres d'entrée de la primitive "response" (réponse); et
6) les paramètres de sortie de la primitive "confirm" (confirmation).

NOTE 3 Les primitives "request", "indication", "response" et "confirm" sont aussi appelées respectivement primitives "requestor.submit", "acceptor.deliver", "acceptor.submit" et "requestor.deliver" (voir ISO/CEI 10731).

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

M le paramètre est obligatoire pour la primitive

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

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

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

S le paramètre est un élément sélectionné.

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

a) une contrainte spécifique au paramètre:

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

b) une indication qu'une certaine note s'applique à l'article:

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

3.8.3.3 Procédures de service

Les procédures sont définies en termes

- d'interactions entre entités d'application par l'échange d'Unités de Données de Protocole d'Application de Bus de Terrain, et
- d'interactions entre un fournisseur de service de couche application et un utilisateur de services de couche application dans le même système par l'invocation de primitives de service de couche application.

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

NOTE La sous-série de normes CEI 61158-5 définit des ensembles de services abstraits. Ce ne sont ni des spécifications de protocoles, ni des spécifications de mises en œuvre, ni des spécifications d'interfaces de programmation. Par conséquent, il existe des restrictions sur la manière dont les procédures de service peuvent être mandatées dans les parties de la sous-série de normes CEI 61158-5. Les aspects des protocoles pouvant varier selon différentes spécifications de protocoles ou différentes mises en œuvre qui instancient les mêmes services abstraits ne conviennent pas pour être inclus dans ces définitions de services, sauf au niveau d'abstraction nécessairement commun à toutes ces expressions.

Par exemple, les moyens par lesquels la paire PDU de demande et PDU de réponse d'un fournisseur de service convient pour une spécification dans une norme de spécifications de protocoles selon la sous-série de normes CEI 61158-6 mais pas dans une norme de définition des services abstraits selon la sous-série de normes CEI 61158-5. De la même manière, les méthodes de mises en œuvres locales par lesquelles des paires de primitives "request-confirm" d'un utilisateur de service ou d'un fournisseur de service conviennent pour une spécification de mises en œuvre ou pour une spécification des interfaces de programmation, mais pas pour une norme de services abstraits ou pour une norme de protocoles, sauf à un niveau d'abstraction nécessairement commun à toutes les applications de la norme de spécification. Dans tous les cas, la définition abstraite n'est pas autorisée à spécifier outre mesure la réalisation d'instanciation plus concrète.

De plus amples informations relatives aux procédures de services conceptuelles d'une mise en œuvre d'un protocole qui effectue les services d'une des définitions de services abstraits selon la sous-série de normes CEI 61158-5 sont disponibles dans la CEI 61158-1, 9.6.

4 Concepts

Les concepts et modèles communs utilisés pour décrire le service de couche application dans la présente norme sont définis en détail à la CEI 61158-1, Article 9.

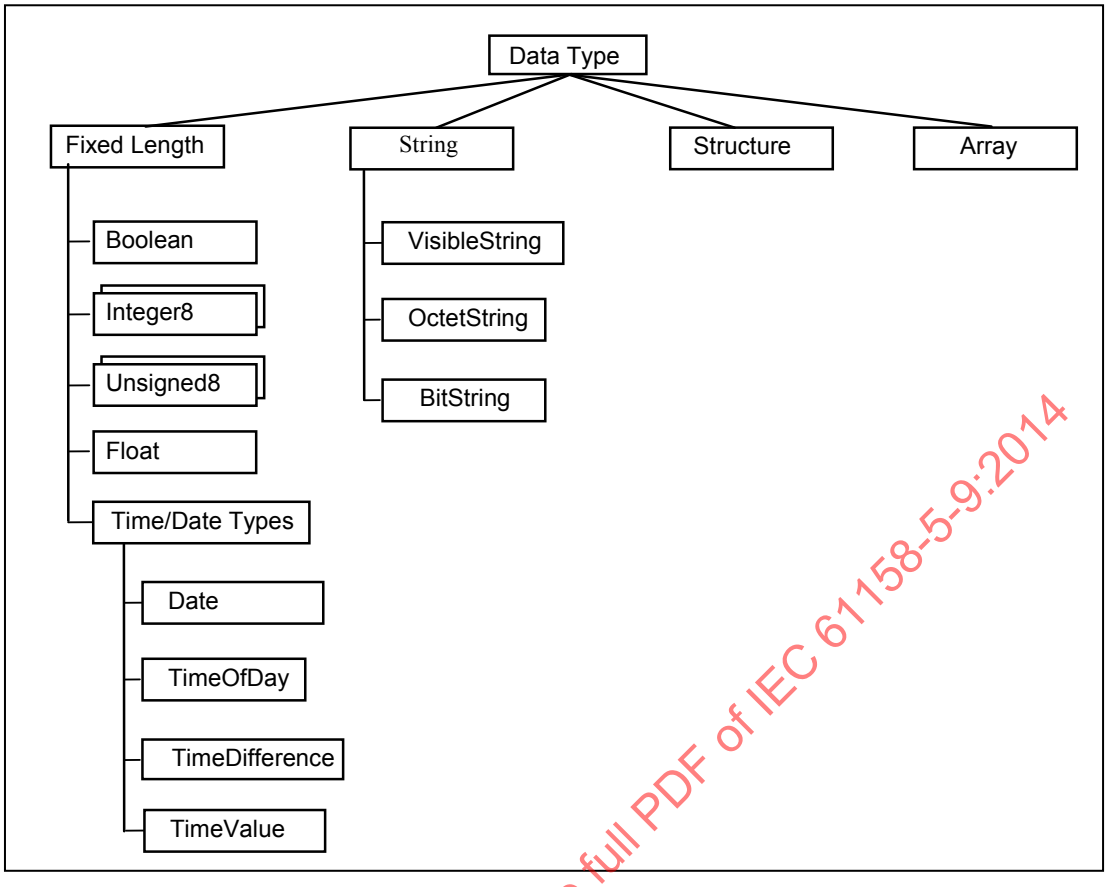
5 ASE des types de données

5.1 Vue d'ensemble

Les types de données de bus de terrain spécifient la syntaxe indépendante vis-à-vis de toute machine pour les données d'application acheminées par les services FAL. La couche d'application de Bus de terrain prend en charge la définition et le transfert des types de données tant de base que construits. Les règles de codage pour les types de données spécifiés dans l'Article 5 sont données dans la CEI 61158-6-9.

Les types de base sont des types atomiques qui ne peuvent pas être décomposés en plus de types élémentaires. Les types construits sont des types composés de types de base et d'autres types construits. Leur complexité et leur profondeur d'imbrication ne sont pas contraintes par la présente norme.

Les types de données sont définis comme des instances de la classe des types de données (voir Figure 1).



Légende

Anglais	Français
Data type	Type de données
Fixed length	Longueur fixe
String	Chaîne
Structure	Structure
Array	Matrice

Figure 1 – Hiérarchie de la classe des types de données

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

Les classes de données de base sont utilisées pour définir des types de données de longueur fixe et chaîne de bits. Les types normalisés issus de l'ISO/CEI 8824-1 sont appelés types de données *simples*. Les autres types de données de base normalisés sont définis spécifiquement pour les applications de Bus de terrain et sont appelés *types spécifiques*.

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

5.1.1 Vue d'ensemble des types de base

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

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

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

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

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

5.1.1.1 Types de Longueur Fixe

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

5.1.2 Vue d'ensemble des types construits

Les types de données construits sont requis pour acheminer intégralement les diverses informations présentes sur le Bus de Terrain. Il existe deux sortes de types construits définis pour la présente norme, les matrices et les structures.

5.1.2.1 Chaînes

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

5.1.2.2 Matrices

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

5.1.2.3 Structures

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

5.1.2.4 Niveau d'imbrication

La présente norme permet que les matrices et les structures contiennent des matrices et des structures. Elle impose une restriction du nombre de niveaux d'imbrication autorisé à un.

5.1.3 Spécification des types de données définis par l'utilisateur

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

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

5.1.4 Transfert de données d'utilisateur

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

Les règles pour coder les données d'utilisateur dans les unités de données du protocole FAL dépendent du type de données. Ces règles sont définies dans la CEI 61158-6-9. Les types de données définis par l'utilisateur pour lesquels il n'y a pas de règles de codage sont transférés sous la forme d'une séquence de longueur variable d'octets. Le format des données dans la chaîne d'octets est défini par l'utilisateur.

5.2 Définition formelle des objets de types de données

5.2.1 Classe "Data Type" (Type de Données)

La classe "Data type" spécifie la racine de l'arbre de la classe des types de données. Sa classe parente "top" indique le sommet de l'arbre de la classe FAL.

ASE FAL:			**ASE DATA TYPE**
CLASSE:			**DATA TYPE**
CLASS ID:			5 (FIXED LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)
CLASSE PARENTE:			TOP
ATTRIBUTS:			
1	(m)	KeyAttribute:	Data type Numeric Identifier
2	(o)	KeyAttribute:	Data type Name
3	(m)	Attribut:	Format (FIXED LENGTH, STRING, STRUCTURE, ARRAY)
4	(c)	Contrainte:	Format = FIXED LENGTH | STRING
4.1	(m)	Attribut:	Octet Length
5	(c)	Contrainte:	Format = STRUCTURE
5.1	(m)	Attribut:	Number of Fields
5.2	(m)	Attribut:	List of Fields
5.2.1	(o)	Attribut:	Field Name
5.2.2	(m)	Attribut:	Field Data type
6	(c)	Contrainte:	Format = ARRAY
6.1	(m)	Attribut:	Number of Array Elements
6.2	(m)	Attribut:	Array Element Data type

5.2.1.1 Attributs

Format

Cet attribut identifie le type de données comme étant de longueur fixe, chaîne, matrice ou structure de données.

Octet Length

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

Number of Fields

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

List of Fields

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

Field Name

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

Field Data type

Cet attribut conditionnel spécifie le type de données du champ. Il est présent lorsque la valeur de l'attribut "format" est "STRUCTURE". Cet attribut peut lui-même spécifier un type de données construit soit en référençant une définition de type de données construit par son id numérique (identificateur numérique), soit en intégrant ici une définition de type de données construit. Pour intégrer une description, la description de l'attribut "Embedded Data type" présentée ci-après est utilisée.

Number of Array Elements

Cet attribut conditionnel définit le nombre d'éléments de type matrice. Les éléments de matrice sont indexés de "0" à "n-1", la taille de la matrice étant de "n" éléments. Cet attribut est présent lorsque la valeur de l'attribut "format" est "ARRAY" (MATRICE).

Array Element Data type

Cet attribut conditionnel spécifie le type de données pour les éléments d'une matrice. Tous les éléments de la matrice ont le même type de données. Il est présent lorsque la valeur de l'attribut "format" est "ARRAY" (MATRICE). Cet attribut peut lui-même spécifier un type de données construit soit en référençant une définition de type de données construit par son id numérique (identificateur numérique), soit en intégrant ici une définition de type de données construit. Pour intégrer une description, la description de l'attribut "Embedded Data type" qui est présentée ci-après, est utilisée.

Embedded Data type Description

Cet attribut est utilisé pour définir de manière récursive les types de données intégrés et ce, au sein d'une structure ou d'une matrice. Le modèle ci-dessous définit son contenu. Les attributs présentés dans le modèle sont définis ci-dessus dans la classe des types de données, à l'exception de l'attribut "Embedded Data type", qui est une référence récursive à cet attribut. Il est utilisé pour définir des éléments imbriqués.

1	(m)	Attribut:	Format(FIXED LENGTH, STRING, STRUCTURE, ARRAY)
2	(c)	Contrainte:	Format = FIXED LENGTH | STRING
2.1	(m)	Attribut:	Data type Numeric ID value
2.2	(m)	Attribut:	Octet Length
3	(c)	Contrainte:	Format = STRUCTURE
3.1	(m)	Attribut:	Number of Fields
3.2	(m)	Attribut:	List of Fields
3.2.1	(m)	Attribut:	Embedded Data type Description
4	(c)	Contrainte:	Format = ARRAY
4.1	(m)	Attribut:	Number of Array Elements
4.2	(m)	Attribut:	Embedded Data type Description

5.3 Types de données définis pour la FAL

5.3.1 Types de longueur fixe

5.3.1.1 Boolean

CLASSE: **Data type**

ATTRIBUTS:

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

Ce type de données exprime un type de données Boolean avec les valeurs TRUE (VRAI) et FALSE (FAUX).

5.3.1.2 Integer8

CLASSE:			**Data type**
ATTRIBUTS:			
1	Data type Numeric Identifier	=	2
2	Data type Name	=	Integer8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

Ce type integer est un nombre binaire en complément à deux avec une longueur égale à un octet.

5.3.1.3 Integer16

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

Ce type integer est un nombre binaire en complément à deux avec une longueur égale à deux octets.

5.3.1.4 Integer32

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

Ce type integer est un nombre binaire en complément à deux avec une longueur égale à quatre octets.

5.3.1.5 Unsigned8

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

Ce type est un nombre binaire. Le bit de poids le plus fort de l'octet de poids le plus fort est toujours utilisé comme le bit de poids le plus fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur égale à un octet.

5.3.1.6 Unsigned16

CLASSE:			**Data type**
ATTRIBUTS:			
1	Data type Numeric Identifier	=	6
2	Data type Name	=	Unsigned16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

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

5.3.1.7 Unsigned32

CLASSE:			**Data type**
ATTRIBUTS:			
1	Data type Numeric Identifier	=	7
2	Data type Name	=	Unsigned32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

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

5.3.1.8 Float

CLASSE:			**Data type**
ATTRIBUTS:			
1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Ce type a une longueur de quatre octets. Le format de float32 est celui défini par l'ISO/CEI/IEEE 60559 comme étant en simple précision ("single precision").

5.3.1.9 Date

CLASSE:			**Data type**
ATTRIBUTS:			
1	Data type Numeric Identifier	=	11
2	Data type Name	=	Date
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

Ce type de données se compose d'années (years), de mois (months), de jours ou de mois (days or months), de jours de la semaine (days of week), d'heures (hours), d'heures normalisées (standard time)/d'heures d'économie d'énergie (daylight savings time), de minutes (minutes), et de millisecondes (milliseconds).

5.3.1.10 TimeOfDay

CLASSE:			**Data type**
ATTRIBUTS:			
1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

Ce type de données est constitué de deux éléments de valeurs unsigned (non signées) et exprime l'heure du jour et la date. Le premier élément est un type de données Unsigned32 et donne l'heure après minuit en millisecondes. Le second élément est un type de données Unsigned16 et donne la date.

5.3.1.11 TimeDifference

CLASSE:			**Data type**
ATTRIBUTS:			

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

Ce type de données est constitué de deux éléments de valeurs unsigned (non signées) qui expriment la différence de temps. Le premier élément est un type de données Unsigned32 qui donne la partie fractionnaire d'un jour en millisecondes. Le second élément facultatif est un type de données Unsigned16 qui donne la différence en jours.

CLASSE: **Data type**

ATTRIBUTS:

1	Data type Numeric Identifier	=	21
2	Data type Name	=	Time Value
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

Ce type simple exprime le temps ou l'écart de temps dans un nombre binaire en complément à deux avec une longueur égale à huit octets. L'unité de temps est 1/32ème de milliseconde.

5.3.1.12 TimeValue

CLASSE: **Data type**

ATTRIBUTS:

1	Data type Numeric Identifier	=	21
2	Data type Name	=	Time Value
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

Ce type simple exprime le temps ou l'écart de temps dans un nombre binaire en complément à deux avec une longueur égale à huit octets. L'unité de temps est 1/32ème de milliseconde.

5.3.2 Types Chaîne

5.3.2.1 VisibleString

CLASSE: **Data type**

ATTRIBUTS:

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

Ce type est défini comme étant le type chaîne de l'ISO/CEI 646.

5.3.2.2 OctetString

CLASSE: **Data type**

ATTRIBUTS:

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

Un OctetString est une séquence ordonnée d'octets, numérotés de 1 à n. Pour les besoins du débat, l'octet 1 de la séquence est appelé le premier octet. La CEI 61158-6-9 définit l'ordre d'émission.

5.3.2.3 BitString

CLASSE: **Data type**

ATTRIBUTS:

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

Ce type de chaîne est défini comme une série de huit bits, numérotés de 0 à 7.

5.4 Spécification des services ASE pour les types de données

Il n'y a pas de services opérationnels définis pour l'objet "type".

5.5 Synthèse des types de données

Cet article comporte une synthèse des types de données définis. Les valeurs de "Class ID" ont été attribuées afin d'être compatibles avec des normes existantes. Le Tableau 1 est trié par "class id" (identificateur de classe).

Tableau 1 – Synthèse des types de données

Class Id	Type de données
1	Boolean
2	Integer8
3	Integer16
4	Integer32
5	Unsigned8
6	Unsigned16
7	Unsigned32

Class Id	Type de données
8	Float
9	VisibleString
10	OctetString
11	Date
12	TimeOfDay
14	BitString
21	TimeValue

6 Spécification du modèle de communication

6.1 Concepts

Les concepts sont ceux du modèle de communication de base décrit dans la CEI 61158-1.

6.2 Paramètres communs

De nombreux services possèdent les paramètres suivants. Au lieu de les définir avec chaque service, les définitions communes suivantes sont fournies.

AREP ID

Ce paramètre fournit des informations suffisantes pour identifier localement la relation entre applications utilisée pour transmettre le service.

VCR

Ce paramètre fournit des informations suffisantes pour identifier localement la VCR et l'AR utilisées pour transmettre le service.

Error Type

Ce paramètre fournit des informations relatives à la raison de l'exécution défaillante du service.

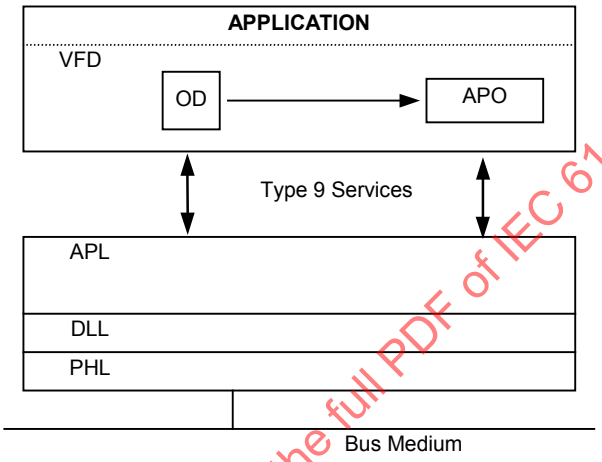
6.3 ASE

6.3.1 ASE de l'appareil de terrain Virtuel

6.3.1.1 Spécification de la classe "Virtual Field Device" (appareil de terrain virtuel)

6.3.1.1.1 Vue d'ensemble de la classe

L'appareil *Virtual Field Device* (VFD) est un modèle abstrait pour la description des données et du comportement d'un Processus d'Application. Les VFD contiennent des APO. Les attributs d'un APO sont décrits par des descriptions d'objets. L'ensemble de descriptions d'objets des VFD est défini comme le Dictionnaire d'Objets *(Object Dictionary)* (OD). Il y a exactement un Dictionnaire d'Objets pour chaque VFD. Des services sont définis pour accéder à un OD de VFD et à ses APO, comme présenté à la Figure 2.



Légende

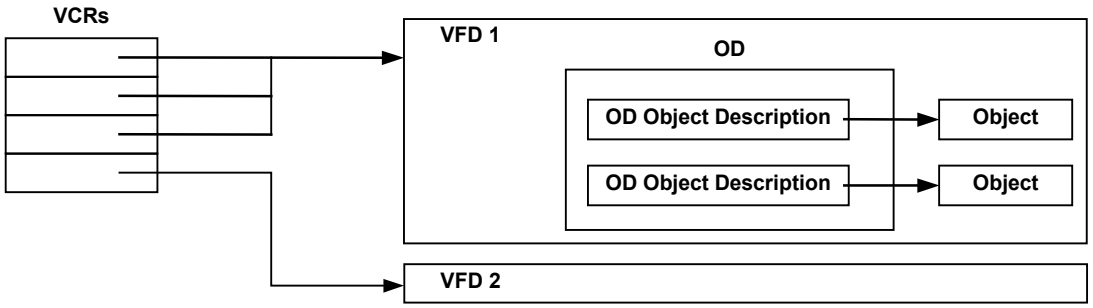
Anglais	Français
Application	Application
Type 9 services	Services de Type 9
Bus medium	Support de bus

Figure 2 – Modèle de VFD

Les services ne définissent pas d'interface concrète pour une mise en œuvre. Ils décrivent de façon abstraite les fonctions qui peuvent être utilisées.

L'application n'est pas abordée dans la présente norme. Il convient de n'indiquer que la manière dont les services décrits de façon abstraite peuvent être mis à disposition de l'application.

L'accès aux objets de VFD est défini par les relations de communication virtuelle *(Virtual Communication Relationships)* (VCR). Plusieurs Objets de VFD peuvent être présents dans un appareil, comme présenté à la Figure 3.



Légende

Anglais	Français
Object	Objet
OD Object Description	Description d'Objets de l'OD
VCRs	VCR

Figure 3 – Modèle abstrait d'un système d'automation (VFD)

6.3.1.1.2 Modèle formel

La classe VFD spécifie les attributs et les services définis pour les processus d'application. Sa classe parente "top" indique le sommet de l'arbre de la classe.

ASE: **ASE VFD**
CLASSE: **VFD**
CLASS ID: **—**
CLASSE PARENTE: **TOP**

1.	(m)	KeyAttribute:	implicit through the VCR
2.	(m)	Attribut:	VendorName
3.	(m)	Attribut:	ModelName
4.	(m)	Attribut:	Revision
5.	(m)	Attribut:	ProfileNumber
6.	(m)	Attribut:	LogicalStatus
7.	(m)	Attribut:	PhysicalStatus
8.	(m)	Attribut:	List of VFD Specific Objects

SERVICES:

1.	(m)	OpsService:	Status
2.	(m)	OpsService:	Unsolicited Status
3.	(m)	OpsService:	Identify

6.3.1.1.3 Attributs

Implicites à la VCR

La VCR qui est entrée dans la Liste des VCR fait référence à l'Objet de VFD attribué.

VendorName

Cet attribut contient le vendeur de l'appareil.

ModelName

Cet attribut spécifie le nom du modèle de l'appareil. La valeur de cet attribut est définie par le vendeur.

Revision

Cet attribut décrit le niveau de révision de l'appareil. La valeur de cet attribut est définie par le vendeur.

ProfileNumber

Cet attribut identifie le profil du VFD. La valeur de profil est définie par une autorité administrative externe. Si l'appareil ne correspond pas à un profil, alors cet attribut a une valeur nulle.

Logical Status

Cet attribut spécifie les informations relatives à l'état des capacités de communication de l'appareil. Ses valeurs sont montrées dans le Tableau 2.

Tableau 2 – Statuts logiques

Statut logique	Signification
Ready for communication (Prêt pour la communication)	Tous les services peuvent être utilisés normalement.
Limited number of services (Nombre de services limité)	Un nombre limité de services peut être pris en charge dans certains cas.
OD-LOADING-NON-INTERACTING (CHARGEMENT-OD-SANS-INTERACTION)	Si le Dictionnaire d'Objets est dans l'état OD-LOADING-NON-INTERACTING, il n'est pas autorisé à exécuter le service InitiatePutOD.
OD-LOADING-INTERACTING (CHARGEMENT-OD-AVEC-INTERACTIONS)	Si le Dictionnaire d'Objets est dans l'état OD-LOADING-INTERACTING, alors toutes les connexions, sauf celle sur laquelle le Service InitiatePutOD a été reçu, sont verrouillées et il convient que tout établissement d'une connexion ultérieure soit rejeté. Sur cette connexion, seuls les services suivants sont autorisés: Initiate PhysWrite Abort GetOD Reject InitiatePutOD Status PutOD Identify TerminatePutOD PhysRead

Physical Status

Cet attribut donne un aperçu grossier de l'état de l'appareil réel. Ses valeurs sont:

– operational (opérationnel)
– partially operational (partiellement opérationnel)
– not operational (non opérationnel)
– needs maintenance (maintenance exigée)

List of VFD Specific Objects

Contient tous les Objets Spécifiques au VDF.

6.3.1.1.4 Services

Status

Ce service est utilisé pour demander l'état visible du réseau en provenance de l'AP.

Unsolicited Status

Ce service est utilisé par l'AP pour rapporter l'état visible de son réseau.

Identify

Ce service est utilisé pour demander des informations relatives au fabricant à l'AP.

6.3.1.2 Spécification des services ASE pour l'appareil de terrain virtuel

6.3.1.2.1 Services pris en charge

Ce paragraphe comporte la définition des services uniques à cet élément ASE. Les services définis pour cet ASE sont:

– Status
– Unsolicited Status

– Identify

6.3.1.2.2 Service Status

6.3.1.2.2.1 Vue d'ensemble du service

L'état de l'appareil/de l'utilisateur est lu avec le service Status.

6.3.1.2.2.2 Primitives de service

Les paramètres de service pour ce service sont présentés dans le Tableau 3.

Tableau 3 – Status

Nom de Paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Result(+)			S	S (=)
Logical Status			M	M (=)
Physical Status			M	M (=)
Local Detail			U	U (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Logical Status

Ce paramètre spécifie la valeur de l'attribut "Logical Status" du VFD.

Physical Status

Ce paramètre spécifie la valeur de l'attribut "Physical Status" du VFD.

Local Detail

Ce paramètre spécifie l'état défini par l'utilisateur de l'application.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.3.1.2.2.3 Procédure de service

La Procédure de Service Confirmé spécifiée à l'Article 4 s'applique à ce service.

6.3.1.2.3 Service Unsolicited status

6.3.1.2.3.1 Vue d'ensemble du service

Le Service Status Notification est utilisé par l'AP pour rapporter spontanément son état.

6.3.1.2.3.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 4.

Tableau 4 – Unsolicited status

Nom de paramètre	Req	Ind
Argument		
VCR	M	M
Destination DL-Address	C	
Source DL-Address		C
LogicalStatus	M	M (=)
Physical Status	M	M (=)
Local Detail	U	U (=)

Argument

L'argument contient les paramètres de la demande de service.

Destination DL-Address

Ce paramètre existe uniquement lorsque l'AREP correspondant le prend en charge et que le Type de Configuration de l'Adresse Distante est FREE (LIBRE). Il indique l'adresse distante à laquelle il convient d'envoyer le service Status Notification.

Source DL-Address

Ce paramètre existe uniquement lorsque l'AREP correspondant le prend en charge et que le Type de Configuration de l'Adresse Distante est FREE (LIBRE). Ce paramètre indique l'adresse distante depuis laquelle le service Status Notification indiqué va être envoyé.

Logical Status

Ce paramètre spécifie la valeur de l'attribut "Logical Status" du VFD.

Physical Status

Ce paramètre spécifie la valeur de l'attribut "Physical Status" du VFD.

Local Detail

Ce paramètre spécifie l'état défini par l'utilisateur de l'application.

6.3.1.2.3.3 Procédure de service

La Procédure de Service non Confirmée spécifiée à l'Article 4 s'applique à ce service.

6.3.1.2.4 Service Identify

6.3.1.2.4.1 Vue d'ensemble du service

Les informations pour identifier un AP sont lues avec ce service.

NOTE L'attribut "ProfileNumber" de l'AP est transmis avec le service Initiate de l'élément ASE de Gestion de Contexte.

6.3.1.2.4.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 5.

Tableau 5 – Identify

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Result(+)			S	S (=)
Vendor Name			M	M (=)
Model Identifier			M	M (=)
Vendor Revision			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Vendor Name

Ce paramètre spécifie la valeur de l'attribut "Vendor Name" du VFD.

Model Identifier

Ce paramètre spécifie la valeur de l'attribut "Model Identifier" du VFD.

Vendor Revision

Ce paramètre spécifie la valeur de l'attribut "Vendor Revision" du VFD.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.3.1.2.4.3 Procédure de service

La Procédure de Service Confirmé spécifiée à l'Article 4 s'applique à ce service.

6.3.2 Elément ASE du Dictionnaire d'Objets (OD)

6.3.2.1 Vue d'ensemble

Chaque Modèle ASE spécifie une ou plusieurs classes d'APO associées en tant qu'ensemble d'attributs et de services.

L'ensemble des attributs pour les APO d'un AP constitue le Dictionnaire d'Objets (OD). Les attributs pour un seul APO constituent sa description d'objets. Pour accéder aux descriptions

d'objets des APO, cet ASE définit des services OD communs. Ces services fonctionnent à l'aide du modèle client/serveur.

Le Dictionnaire d'Objets (OD) contient les Descriptions d'Objets des Objets de Communication suivants:

- DataType (Type de Données)
- Domain (Domaine)
- SimpleVariable (Variable Simple)
- Record (Enregistrement)
- Event (Evénement)
- DataTypeStructureDescription (Description de la Structure des Types de Données)
- ProgramInvocation (Invocation de Programme)
- Array (Matrice)
- VariableList (Liste de Variable)

Une description d'objets "Dictionnaire d'Objets" (description d'objets de l'OD) est attribuée de manière unique au Dictionnaire d'Objets. La description d'objets de l'OD contient des informations relatives à la structure du Dictionnaire d'Objets.

Les descriptions d'objets sont marquées avec un indice unique de type Unsigned16.

Un nom de type chaîne visible peut également être attribué aux objets suivants:

- Domain (Domaine)
- SimpleVariable (Variable Simple)
- Record (Enregistrement)
- Event (Evénement)
- ProgramInvocation (Invocation de Programme)
- Array (Matrice)
- VariableList (Liste de Variables)

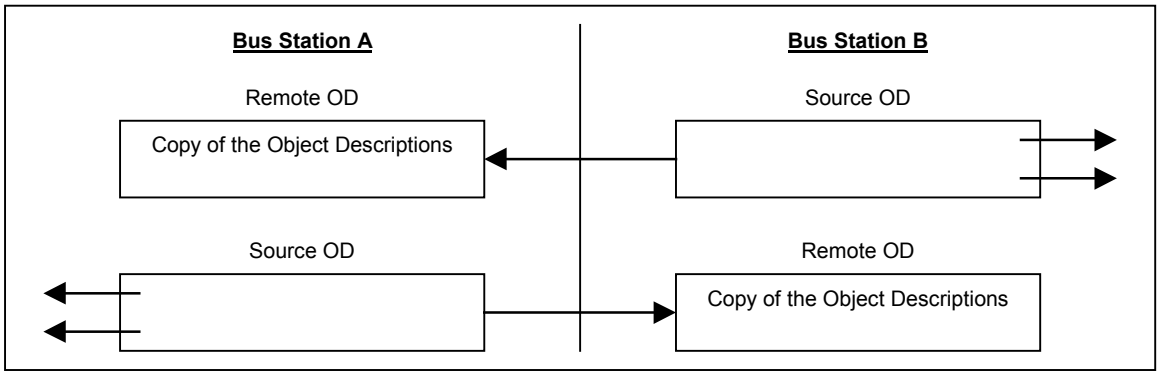
L'indice ou le nom font office de clé à l'objet et à la description de l'objet.

Les services adressent en principe l'objet du serveur, car l'objet réel existe ici.

Avant qu'un partenaire de communication distant puisse accéder à un objet, il nécessite de connaître sa description. L'adresse logique (Indice) ou le nom peuvent être utilisés, lorsque l'accès des objets est effectué à distance.

Les descriptions d'objets peuvent être spécifiées lors de la configuration du système, mais elles peuvent également être transférées entre les stations à tout moment ultérieurement.

La définition de la description d'objets se déroule à la station dans laquelle l'objet existe réellement (OD Source). La longueur d'une description d'objets peut être différente pour les objets individuels. Les partenaires de communication conservent une copie intégrale ou partielle de la Description de l'Objet Distant (OD distant). Ceci est illustré à la Figure 4.



Légende

Anglais	Français
Bus Station A	Station de Bus A
Remote OD	OD distant
Copy of the Object Descriptions	Copie des Descriptions d'Objets
Source OD	OD source
Bus Station B	Station de Bus B

Figure 4 – OD source/OD distant

Par conséquent, chaque station peut posséder un OD Source pour les objets de communication existants localement et un ou plusieurs OD Distants. Les OD Distants sont spécifiques à la connexion, c'est-à-dire que pour chaque connexion, un autre OD Distant peut être valide.

Le modèle d'OD fournit des services correspondants pour la lecture et l'écriture d'OD Sources. Si un OD Source est modifié par un partenaire de communication distant, il convient alors que cela soit notifié à l'ensemble des autres partenaires de communication qui utilisent la même base de données (OD Source). Pour ce faire, l'utilisateur libère les connexions à tous les autres partenaires de communication. Après que l'OD ait été modifié, les connexions peuvent être rétablies. Lors de l'établissement de la connexion, le numéro de la nouvelle version (Version de l'OD) est fourni à chaque partenaire de communication pour leur indiquer que l'OD Source correspondant a été modifié. L'ajout de descriptions d'objets est autorisé sans libération de connexion.

La structure de l'OD décrite comme suit n'impose pas de mise en œuvre concrète de l'OD. Elle définit seulement la structure des descriptions d'objets qui sont transmises par le biais du bus. L'existence d'un OD n'est même pas nécessaire si les informations requises peuvent être obtenues d'une autre manière (par exemple, par un algorithme). Les attributs et les syntaxes de transfert détaillés sont spécifiés dans les modèles appropriés.

Le Dictionnaire d'Objets (OD) est divisé en plusieurs parties, comme présenté au Tableau 6:

– La description d'Objets du "Dictionnaire d'Objets" (description d'objets de l'OD) comporte les informations structurelles de l'OD.
– Le Dictionnaire des Types Statique (ST-OD) comporte des objets "Data Types" et "Data type Structure Description".
– Le Dictionnaire d'Objets Statique (S-OD) contient des descriptions d'objets de "Simple Variables", "Arrays", "Records", "Domains" et d'"Events".
– La Liste Dynamique des Listes de Variables (DV-OD) comporte des descriptions d'objets de "Variable Lists".
– La Liste Dynamique des Invocations de Programme (DP-OD) comporte des descriptions d'objets de "Program Invocations".

Tableau 6 – Structure du dictionnaire d'objets

Indice	Dictionnaire d'Objets (OD)
0	Description d'Objets de l'OD (OD-ODES)
1 ... i	Liste Statique des Types (ST-OD)
k ... n	Dictionnaire d'Objets Statique (S-OD)
p ... t	Liste Dynamique des Listes de Variables (DV-OD)
v ... z	Liste Dynamique des Invocations de Programme (DP-OD)

6.3.2.1.1 Liste Statique des Types (ST-OD)

La Liste Statique des Types (ST-OD) contient les descriptions d'objets des "Data Types" et des "Data Type Structure Descriptions" pour les Objets "Variable Access" (Accès aux Variables) du S-OD, comme présentée dans le Tableau 7.

Tableau 7 – Structure de la liste statique des types

Indice	Liste Statique des Types (ST-OD)
1 ... c	Contient des objets: "Data Types"
d ... i	Contient des objets: "Data Types" et "Data Type Structure Descriptions"

Exactement une description d'objets est attribuée à chaque indice. Cette attribution est statique et ne peut pas être supprimée. Le ST-OD peut être chargé par le partenaire distant.

La Liste Statique des Types commence toujours par l'indice 1 dans le Dictionnaire d'Objets. Il convient que les descriptions d'objets restantes suivent avec des indices croissants. Le nombre d'attributions "indice <=> description d'objets" est saisi comme la longueur du ST-OD dans le Dictionnaire d'Objets.

Le ST-OD est divisé en deux segments. Le premier segment contient les descriptions d'objets des "Data Types" normalisés (Types de Données Standard) et commence par l'indice 1. Le second segment suit immédiatement et contient des descriptions d'objets de "Data Types" et de "Data Type Structure Descriptions", qui peuvent être définies individuellement. Ce segment peut être spécifié ultérieurement par des groupes d'utilisateurs à l'aide de profils.

6.3.2.1.2 Dictionnaire d'Objets Statique (S-OD)

Le Dictionnaire d'Objets Statique (S-OD) contient des descriptions d'objets de "Simple Variables", "Arrays", "Records", "Domains" et "Events" comme présenté au Tableau 8.

Tableau 8 – Structure du dictionnaire d'objets statique

Indice	Dictionnaire d'objets statique (S-OD)
k ... n	contient des objets: "Simple Variables", "Arrays", "Records", "Domains" et "Events"

Une description d'objets est attribuée à chaque indice. Le premier indice, associé à une description d'objets, est entré dans la description d'objets du Dictionnaire d'Objets. Il convient que les descriptions d'objets restantes suivent avec des indices croissants. Le nombre d'attributions "indice <=> description d'objets" est saisi comme la longueur du S-OD dans la description d'objets du Dictionnaire d'Objets.

En plus de l'attribution "indice <=> description d'objets", une attribution supplémentaire "nom <=> description d'objets" peut exister. La longueur du nom peut être de 0 à 32 octets. Elle est entrée dans le Champ "Name Length" (Longueur du Nom) de la description d'objets de l'OD. Si ce champ contient un "0", aucun nom n'est présent. Les objets définis dans le S-OD peuvent être fournis avec des droits d'accès. Il apparaît dans le champ "AccessProtectionSupported" de la description d'objets de l'OD si les Droits d'Accès sont possibles ou non.

Une description d'objets dans le S-OD contient une extension, qui peut contenir d'autres définitions spécifiques aux objets.

6.3.2.1.3 Liste dynamique des listes de variables (DV-OD)

La Liste Dynamique des Listes de Variables (DV-OD) contient la description d'objets de "Variable List", comme présentée au Tableau 9.

Tableau 9 – Structure de la liste dynamique des listes de variables

Indice	Liste Dynamique des Listes de Variables (DV-OD)
p ... t	contient des objets: "Variable List"

Un objet "Variable List" et sa description d'objets dans le DV-OD est créé dynamiquement avec le service DefineVariableList et peut être supprimé avec le service DeleteVariableList. Dès la création de l'objet, des droits d'accès peuvent y être attribués. De plus, il est possible de créer ces objets en chargeant l'OD dans son intégralité.

Le premier indice, qui possède la description d'objets de "Variable List" qui lui est attribuée, et le nombre d'indices disponibles sont saisis dans la description d'objets de l'OD.

Les informations fondamentales sont saisies dans la description d'objets de "Variable List", par exemple les droits d'accès, le nombre d'objets "Variable Access" et les adresses logiques (indices) des objets "Variable Access" dans le S-OD.

En plus de l'attribution "indice <=> description d'objets", une attribution supplémentaire "nom <=> description d'objets" peut exister.

6.3.2.1.4 Liste dynamique des Invocations de Programme (DP-OD)

La Liste Dynamique des Invocations de Programme (DV-OD) contient les descriptions d'objets des Objets "Program Invocation", comme présentée au Tableau 10.

Tableau 10 – Structure de la liste dynamique des invocations de programme

Indice	Liste Dynamique des Invocations de Programme (PD-OD)
v ... w	Invocations de Programme Prédéfinies
x ... z	Invocations de Programme Dynamiques

Un objet "Program Invocation" et sa description d'objets dans le DP-OD sont créés dynamiquement avec le service Create Program Invocation et supprimés avec le service Delete Program Invocation. Dès la création de l'objet, des droits d'accès peuvent y être attribués. De plus, il est possible de créer ces objets en chargeant l'OD tout entier.

Le premier indice, qui possède la description d'objets de "Program Invocation" qui lui est attribuée, et le nombre d'indices disponibles sont saisis dans la description d'objets de l'OD.

La description d'objets de "Program Invocation" contient des informations telles que les droits d'accès et le nombre d'objets "Domain" et leurs adresses logiques (indices).

En plus de l'attribution "indice <=> description d'objets", une attribution supplémentaire "nom <=> description d'objets" peut exister.

Le DP-OD peut contenir un segment avec des Invocations de Programme préconfigurées.

6.3.2.2 Spécifications de la classe OD

6.3.2.2.1 Vue d'ensemble de la classe

Chaque description d'un objet contient un indice, un code objet, les attributs de l'objet, des références à l'objet réel spécifiques au système, et, le cas échéant, un nom et une extension.

L'objet est adressé par l'indice et identifié par la description de l'objet. Le code objet est l'identificateur de l'objet et indique la classe à laquelle appartient cet objet. Les autres attributs de l'objet sont spécifiques à l'objet. Le contenu du champ d'extension est spécifique à l'application.

L'adresse locale permet l'adressage interne de l'objet. En plus de l'indice, un nom pour l'adressage de l'objet peut également être utilisé de manière facultative. L'extension contient une information relative à la longueur pour l'extension et, de manière facultative, d'autres attributs de l'objet. La valeur "0" pour la longueur signifie: pas d'autres attributs de l'objet.

6.3.2.2.2 Spécification de la classe "OD description" (Description de l'OD)

6.3.2.2.2.1 Modèle formel

La structure du Dictionnaire d'Objets est décrite par le premier article dans l'OD. Elle possède l'indice "0" dans le Dictionnaire d'Objets. En lisant la description d'objets de l'OD, la description de la structure et le numéro de la version de l'OD sont rendus disponibles.

ASE: **ASE du Dictionnaire d'Objets**
CLASSE: **OD Description**
CLASS ID: **—**
CLASSE PARENTE: **TOP**
ATTRIBUTS:
1. (m) KeyAttribute: OD Index

2. (m) Attribut: ROM/RAM Flag
3. (m) Attribut: Name Length
4. (m) Attribut: Access Protection Supported
5. (m) Attribut: Version OD
6. (m) Attribut: Local Address OD-ODES
7. (m) Attribut: ST-OD Length
8. (m) Attribut: LocalAddress ST-OD
9. (m) Attribut: First Index S-OD
10. (m) Attribut: S-OD Length
11. (m) Attribut: Local Address S-OD
12. (m) Attribut: First Index DV-OD
13. (m) Attribut: DV-OD Length
14. (m) Attribut: LocalAddress DV-OD
15. (m) Attribut: First Index DP-OD
16. (m) Attribut: DP-OD Length
17. (m) Attribut: Local Address DP-OD

SERVICES:

1. (m) OpsService: Get OD
2. (m) OpsService: Initiate Put OD
3. (m) OpsService: Put OD
4. (m) OpsService: Terminate Put OD

6.3.2.2.2.2 Attributs

OD Index

Indice OD de la Description de l'Objet OD, ici toujours = 0.

ROM/RAM Flag

Cet attribut de type Boolean décrit si, oui ou non, les modifications dans l'OD sont autorisées.

false <=> aucune modification dans l'OD n'est permise

true <=> les modifications dans l'OD sont permises

Name Length

Le champ indique la longueur du nom et ne peut prendre que les valeurs comprises entre 0 et 32.

0 <=> aucun nom n'est utilisé

Access Protection Supported

Cet attribut indique si les droits d'accès pour le mot de passe, les groupes d'accès et tous les partenaires de communication sont pris en charge (true) (vrai), ou si, oui ou non, l'accès à tous les objets est autorisé pour chaque partenaire de communication (false) (faux).

Version OD

Indique la version de l'OD.

Local Address OD-ODES

Cet attribut est une référence à la description de l'objet de l'OD réel spécifique au système et permet l'adressage interne. Si aucune représentation de ce type n'est utilisée, cet attribut est mis sur la valeur hexadécimale FFFFFFFF.

ST-OD Length

Cet attribut indique le nombre maximal d'entrées disponibles dans la Liste Statique des Types.

NOTE Le premier indice disponible de la Liste Statique des Types est toujours = 1, par conséquent l'indice le plus élevé disponible dans la Liste Statique des Types est égal à sa longueur.

Local Address ST-OD

Cet attribut est une référence au ST-OD réel spécifique au système et permet l'adressage interne. Si aucune représentation de ce type n'est utilisée, cet attribut est mis sur la valeur hexadécimale FFFFFFFF.

First Index S-OD

Le premier indice disponible dans le Dictionnaire d'Objets Statique.

S-OD Length

Cet attribut indique le nombre maximal d'entrées disponibles dans le Dictionnaire d'Objets Statique.

NOTE L'indice le plus élevé disponible est S-OD-Length = FirstIndex-S-OD + S-OD-Length - 1 (effectué)

Local Address S-OD

Cet attribut est une référence au S-OD réel spécifique au système et permet l'adressage interne. Si aucune représentation de ce type n'est utilisée, cet attribut est mis sur la valeur hexadécimale FFFFFFFF.

First Index DV-OD

Premier indice disponible dans la Liste Dynamique des Listes de Variables.

0 <=> Aucune Liste Dynamique des Listes de Variables.

DV-OD Length

Cet attribut indique le nombre maximal d'entrées disponibles dans la Liste Dynamique des Listes de Variables.

NOTE L'indice le plus élevé disponible est DV-OD-Length = FirstIndex-DV-OD + OD-OD-Length – 1.

Local Address DV-OD

Cet attribut est une référence à la DV-OD réelle spécifique au système et permet l'adressage interne. Si aucune représentation de ce type n'est utilisée, cet attribut est mis sur la valeur hexadécimale FFFFFFFF.

First Index DP-OD

Premier indice disponible dans la Liste Dynamique des Invocations de Programme.

0 <=> Aucune Liste Dynamique des Invocations de Programme.

DP-OD Length

Cet attribut indique le nombre maximal d'entrées disponibles dans la Liste Dynamique des Invocations de Programme.

NOTE L'indice le plus élevé disponible est DP-OD-Length = FirstIndex-DP-OD + DP-OD-Length – 1.

Local Address DP-OD

Cet attribut est une référence à la DP-OD réelle spécifique au système et permet l'adressage interne. Si aucune représentation de ce type n'est utilisée, cet attribut est mis sur la valeur hexadécimale FFFFFFFF.

6.3.2.2.2.3 Services

Tous les services définis pour cette classe sont obligatoires.

Get OD

Ce service confirmé est utilisé pour lire les entrées de l'OD.

Initiate Put OD

Ce service confirmé est utilisé pour commencer le processus d'écriture des entrées de l'OD.

Put OD

Ce service confirmé est utilisé pour écrire les entrées de l'OD.

Terminate Put OD

Ce service confirmé est utilisé pour mettre fin au processus d'écriture des entrées de l'OD.

6.3.2.2.2.4 Dictionnaire d'objets vide

Le Dictionnaire d'Objets est chargeable. Par conséquent, si un Dictionnaire d'Objets n'a pas encore été chargé, il convient qu'au moins un Dictionnaire d'Objets vide existe.

Un Dictionnaire d'Objets vide est défini en préconfigurant les attributs de la description d'objets de l'OD comme illustré au Tableau 11:

Tableau 11 – Dictionnaire d'objets vide

Attribut	Valeur
ROM/RAM-Flag	True
NameLength	0
AccessProtectionSupported	False
VersionOD	0
ST-OD-Length	0
FirstIndex-S-OD	0
S-OD-Length	0
FirstIndex-DV-OD	0
DV-OD-Length	0
FirstIndex-DP-OD	0
DP-OD-Length	0

6.3.2.2.3 Modèle formel des objets d'OD

6.3.2.2.3.1 Modèle formel

La structure du Dictionnaire d'Objets est décrite par le premier article dans l'OD. Elle possède l'indice "0" dans le Dictionnaire d'Objets. En lisant la description d'objets de l'OD, la description de la structure et le numéro de la version de l'OD sont rendus disponibles.

ASE: **ASE du Dictionnaire d'Objets**
CLASSE: **OD Object**
CLASS ID: **—**
CLASSE PARENTE: **TOP**
ATTRIBUTS:

1. (m) KeyAttribute: implicit by VFD
2. (m) Attribut: OD object description
3. (m) Attribut: List of object description
4. SERVICES:

1. (m) OpsService: Get OD
2. (m) OpsService: Initiate Put OD
3. (m) OpsService: PutOD
4. (m) OpsService: Terminate Put OD

6.3.2.2.3.2 Attributs

implicit by VFD

La VCR qui est entrée dans la liste de VCR fait référence au Dictionnaire d'Objets attribué.

OD object description

Contient la description d'objets du Dictionnaire d'Objets.

List of object description

Contient les descriptions d'objets des objets suivants:

Domain (Domaine)
ProgramInvocation (Invocation de Programme)
SimpleVariable (Variable Simple)
Array (Matrice)
Record (Enregistrement)
VariableList (Liste de Variable)
DataType (Type de Données)
DataTypeStructureDescription (Description de la Structure des Types des Données)
Event (Evénement)

6.3.2.2.3.3 Services

Tous les services définis pour cette classe sont obligatoires.

Get OD

Ce service confirmé est utilisé pour lire les entrées de l'OD.

Initiate Put OD

Ce service confirmé est utilisé pour commencer le processus d'écriture des entrées de l'OD.

Put OD

Ce service confirmé est utilisé pour écrire les entrées de l'OD.

Terminate Put OD

Ce service confirmé est utilisé pour mettre fin au processus d'écriture des entrées de l'OD.

6.3.2.3 Services du modèle de dictionnaire d'objets

6.3.2.3.1 Services pris en charge

Les services OD sont spécifiés pour lire ou écrire les descriptions d'objets des objets "Domain", "Program Invocation", "Simple Variable", "Array", "Record", "Variable List", "Event", "Object Dictionary", "Data type" et "Data type Structure Description". Les attributs d'un APO qui peuvent être écrits sont spécifiés comme faisant partie de la définition des attributs.

Les modifications inattendues des définitions d'objets au sein d'un AP pouvant affecter le fonctionnement d'un AP, les services Initiate Put OD et Terminate Put OD sont définis. Le service Initiate Put OD est utilisé pour demander à l'AP distant l'autorisation de modifier l'OD. Le service Terminate Put OD est utilisé pour indiquer à l'AP distant que les modifications effectuées sur son OD sont terminées et qu'il peut reprendre un fonctionnement normal.

Avec le service GetOD, une ou plusieurs descriptions d'objets sont lues. Avec le service PutOD, une ou plusieurs descriptions d'objets sont écrites. Le nombre de descriptions d'objets, transférables avec un service PutOD ou GetOD, dépend de leur longueur et de la taille maximale disponible des unités PDU.

Avec les services PutOD, une ou plusieurs descriptions d'objets sont écrites. Avant de commencer une modification de l'OD, un utilisateur peut prendre des mesures appropriées (par exemple, mettre un processus dans un état sûr). La communication ne soumet pas à l'essai la plausibilité du nouveau Dictionnaire d'Objets.

L'écriture dans l'OD est effectuée avec la séquence service InitiatePutOD, un ou plusieurs services PutOD et un service TerminatePutOD. Pour ce faire, une distinction est faite entre un chargement sans interaction et un chargement avec interactions (OD-LOADING-NON-INTERACTING (CHARGEMENT-OD-SANS-INTERACTION) et OD-LOADING-INTERACTING (CHARGEMENT-OD-AVEC-INTERACTIONS)). Le chargement avec interaction effectue une distinction entre un nouveau chargement (intégral) et un après-chargement (partiel). Lorsque cela s'avère nécessaire, de nouveaux objets de communication et leurs diagrammes d'états sont créés avec le service TerminatePutOD.

Chargement sans interaction (OD-LOADING-NON-INTERACTING):

Le client communique au serveur avec le paramètre Conséquence = 0, si les modifications ou extensions prévues n'interagissent pas avec les autres partenaires de communication. Dans ce cas, les VCR sont continues. Par exemple, des modifications sans interaction sont

– l'ajout de nouvelles descriptions d'objets dans le segment statique,
– la suppression de descriptions d'objets privés pour les stations qui ne sont plus dans le système de communication.

La communication ne vérifie pas si l'écriture sur l'OD s'effectue réellement sans interaction.

Le chargement sans interaction est initié avec le service InitiatePutOD avec le paramètre Consequence = 0. Les descriptions d'objets peuvent ensuite être chargées avec PutOD. Tant que le processus de chargement n'est pas terminé avec TerminatePutOD, chaque InitiatePutOD supplémentaire est rejeté.

Chargement avec interactions (OD-LOADING-INTERACTING):

Le chargement avec interactions est initié avec le service InitiatePutOD avec le Paramètre Consequence = 1 (après chargement) ou Consequence = 2 (nouveau chargement). Dans ce cas, l'utilisateur peut abandonner toutes les VCR actives, sauf celle sur laquelle le service InitiatePutOD est reçu à ce VFD. Ensuite, la description d'objets peut être chargée avec le service PutOD. C'est également à l'utilisateur client de saisir le numéro de la nouvelle version dans l'OD. Le processus de chargement est terminé avec le service TerminatePutOD. Ensuite, toutes les VCR peuvent être rétablies. Avec le rétablissement des VCR, le numéro de la nouvelle version de l'OD est transféré. Les partenaires de communication peuvent ensuite reconnaître que l'OD correspondant a été modifié. En revanche, pour l'après-chargement, seules des parties de l'OD sont chargées, l'OD existant est supprimé et, par la suite, il est de nouveau chargé intégralement dès réception d'un service InitiatePutOD avec Consequence = 2.

6.3.2.3.2 Service Get OD

6.3.2.3.2.1 Vue d'ensemble du service

Le service fait une distinction entre une forme courte et une forme longue. La forme longue du service GetOD est facultative. Pour lire une seule description d'objets, il convient que l'indice ou le nom soit donné.

Pour lire plusieurs ou toutes les descriptions d'objets, il convient que l'indice de la première description d'objets exigée (StartIndex) (Indice de Départ) soit donné. Pour lire l'intégralité de l'OD, il convient que le service GetOD soit utilisé de manière répétée.

6.3.2.3.2.2 Paramètres du service

Les paramètres de service pour ce service sont présentés dans le Tableau 12.

Tableau 12 – Paramètre du service Get OD

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
All Attributes	M	M (=)		
Access Specification	M	M (=)		
Index	S	S (=)		
Variable Name	S	S (=)		
Variable List Name	S	S (=)		
Domain Name	S	S (=)		
PI Name	S	S (=)		
Event Name	S	S (=)		
Start Index	S	S (=)		
Result(+)			S	S (=)
List of Object Description			M	M (=)
More Follows			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

All Attributes

Ce paramètre indique s'il convient de transférer la description d'objets sous la forme courte (false) (faux) ou sous la forme longue (true) (vrai).

Les attributs d'objet suivants ne sont pas contenus sous la forme courte:

- Password
- AccessGroups
- AccessRights
- LocalAddress
- Name
- LocalAddress-OD-ODES
- LocalAddress-ST-OD
- LocalAddress-S-OD
- LocalAddress-DV-OD
- LocalAddress-DP-OD
- Extension

Access Specification

Ce paramètre indique la description d'objets faisant l'objet d'un accès.

Index

Indice de la description d'objets.

Variable Name

Nom d'une variable.

Variable List Name

Nom d'une Liste de Variables.

PI Name

Nom d'une Invocation de Programme.

Domain Name

Nom d'un Domaine.

Event Name

Nom d'un Evénement.

StartIndex

Indice, à partir duquel il convient de lire les descriptions d'objets. Lorsque cette option est sélectionnée, le VFD répondant envoie des descriptions d'objets des indices supérieurs à StartIndex dans l'ordre croissant. Lorsque toutes les descriptions d'objets demandées ne sont pas envoyées dans la réponse, il convient que l'indicateur MoreFollows soit mis en place.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

List of object description

Pour un accès logique et pour un accès par nom, une ou plusieurs descriptions d'objets complètes. Le nombre dépend de la taille maximale possible des unités PDU et de la longueur des descriptions d'objets uniques. Si les droits d'accès ne sont pas pris en charge (Access Protection = false), les attributs "Password", "Access Groups" et "Access Rights" sont dépourvus de sens.

More Follows

Indique si d'autres descriptions d'objets existent après la lecture de plusieurs descriptions d'objets en sélectionnant StartIndex pour Access Specification. Ce paramètre est mis sur "true" (vrai) lorsque le VFD répondant possède au moins une description d'objets dont l'indice est supérieur à celui de la dernière description d'objets dans la réponse. Il est mis sur "false" (faux) lorsque le VFD répondant ne possède pas plus de descriptions d'objets. Pour l'accès à des descriptions d'objets uniques en sélectionnant d'autres options pour Access Specification (Spécification de l'Accès), descriptions, ce paramètre est toujours mis sur "false" (faux).

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.3.2.3.2.3 Procédure de service

La Procédure de Service Confirmé spécifiée à l'Article 4 s'applique à ce service.

Si l'utilisateur est capable d'accéder aux descriptions d'objets spécifiées, et si elles peuvent être acheminées dans une seule unité APDU, elles sont retournées dans la primitive "response" (+).

Si elles ne peuvent pas être acheminées dans une seule unité APDU, l'utilisateur retourne celles qui peuvent l'être et met en place l'indicateur "more" dans la primitive "response" (+). Le demandeur est ensuite chargé de soumettre des demandes supplémentaires en identifiant les attributs restants à retirer.

Si l'utilisateur est incapable d'obtenir la description pour le premier objet de la liste, le service échoue et l'utilisateur fournit une primitive "response" (-) en indiquant la raison.

6.3.2.3.3 InitiatePutOD

6.3.2.3.3.1 Vue d'ensemble du service

Le service InitiatePutOD lance le processus de chargement de l'OD sans interaction ou avec interactions.

6.3.2.3.3.2 Paramètres du service

Les paramètres de service pour ce service sont présentés dans le Tableau 13.

Tableau 13 – Paramètres du service Initiate put OD

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Consequence	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

Consequence

Ce paramètre indique si, oui ou non, les modifications prévues sont sans interaction pour les autres partenaires de communication, avec interaction avec le nouveau chargement partiel de l'OD, ou avec interaction avec le nouveau chargement intégral de l'OD.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.3.2.3.3.3 Procédure de service

La Procédure de Service Confirmé spécifiée à l'Article 4 s'applique à ce service.

6.3.2.3.4 Service Put OD

6.3.2.3.4.1 Vue d'ensemble du service

Avec le service PutOD, une ou plusieurs descriptions d'objets sont écrites dans l'OD.

6.3.2.3.4.2 Paramètres du service

Les paramètres de service pour ce service sont présentés dans le Tableau 14.

Tableau 14 – Paramètres du service Put OD

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
List of Object Description	M	M (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)
NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.				

Argument

L'argument contient les paramètres de la demande de service.

List of Object Description

Une ou plusieurs descriptions d'objets qu'il convient de saisir dans l'OD. Si les droits d'accès ne sont pas pris en charge (Access Protection = false), les attributs "Password", "Access Groups" et "Access Rights" sont dépourvus de sens.

Une description d'objets peut également être supprimée avec le service PutOD. Dans ce cas, une description d'objets vide avec la structure suivante est transférée:

– Indice de la description d'objets à supprimer
– Code Objet = Null Object.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

6.3.2.3.4.3 Procédure de service

La Procédure de Service Confirmé spécifiée à l'Article 4 s'applique à ce service.

6.3.2.3.5 Service Terminate put OD

6.3.2.3.5.1 Vue d'ensemble du service

Le service Terminate Put OD met fin au processus de chargement de l'OD. Les objets avec leurs diagrammes d'états sont générés.

6.3.2.3.5.2 Paramètres du service

Les paramètres de service pour ce service sont présentés dans le Tableau 15.

Tableau 15 – Paramètres du service Terminate put OD

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)
Index			C	C (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Index

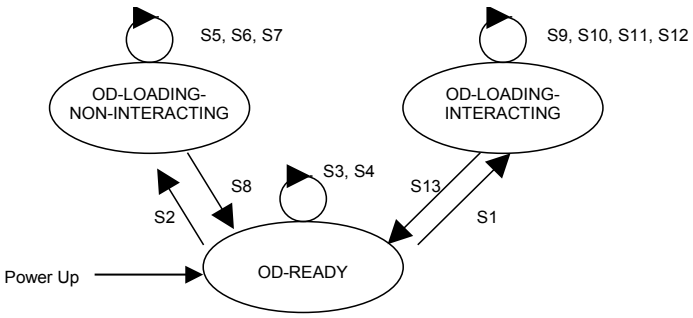
Ce paramètre indique l'Indice de l'objet, pour lequel la génération a échoué.

6.3.2.3.5.3 Procédure de service

La Procédure de Service Confirmé spécifiée à l'Article 4 s'applique à ce service.

6.3.2.3.6 Diagramme d'états des services Put OD

Le diagramme d'états apparaît à la Figure 5.



Légende

Anglais	Français
Power up	Mise sous tension

OD-LOADING-NON-INTERACTING	OD-LOADING-NON-INTERACTING
OD-LOADING-INTERACTING	OD-LOADING-INTERACTING (CHARGEMENT-
OD-READY	OD-READY (OD-PRET)

Figure 5 – Diagramme d'états de Put OD

6.3.2.3.6.1 Etats

OD-LOADING-NON-INTERACTING (CHARGEMENT-OD-SANS-INTERACTION)

Dans cet état, l'exécution du service InitiatePutOD n'est pas autorisée. Dans cet état, le service PutOD sur le DV-OD ou le DP-OD n'est pas autorisé.

OD-LOADING-INTERACTING (CHARGEMENT-OD-AVEC-INTERACTIONS)

Sauf pour la connexion sur laquelle le service InitiatePutOD a été reçu, toutes les connexions sont verrouillées et il convient de rejeter l'établissement d'une autre connexion. Sur cette connexion, seuls les services suivants sont autorisés:

- Initiate
- Abort
- Reject
- Status
- Identify
- PhysRead
- PhysWrite
- GetOD
- InitiatePutOD
- PutOD
- TerminatePutOD

OD-READY (OD-PRET)

Dans cet état, toutes les connexions et tous les services peuvent être utilisés normalement.

6.3.2.3.6.2 Transitions d'états

Le Tableau 16 spécifie les transitions d'états relatives à PutOD.

Tableau 16 – Transitions d'états de Put OD

#	Etat actuel	Evénements	Actions	Etat suivant
S1	OD-READY	InitiatePutOD.ind && consequence <> loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-INTERACTING
S2	OD-READY	InitiatePutOD.ind && consequence = loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-NON-INTERACTING

S3	OD-READY	PutOD.ind	PutOD.rsp(-){
			ErrorType = object-state-conflict
			}
				OD-READY

S4	OD-READY	TerminatePutOD.ind	TerminatePutOD.rsp(-){
			ErrorType = object-state-conflict
			}
				OD-READY

S5	OD-LOADING-NON-INTERACTING	InitiatePutOD.ind	InitiatePutOD.rsp(-){
			ErrorType = object- state-conflict
			}
				OD-LOADING-NON-INTERACTING

S6	OD-LOADING-NON-INTERACTING	PutOD.ind	PutOD.rsp(+){ }	OD-LOADING-NON-INTERACTING

S7	OD-LOADING-NON-INTERACTING	TerminatePutOD.ind
		&& OD is not usable
			TerminatePutOD.rsp(-){
			ErrorType = operational-problem
			}
				OD-LOADING-NON-INTERACTING

S8	OD-LOADING-NON-INTERACTING	TerminatePutOD.ind && OD is okay	TerminatePutOD.rsp(+){ }	OD-READY
S9	OD-LOADING-INTERACTING	InitiatePutOD.ind && consequence <> loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-INTERACTING

S10	OD-LOADING-INTERACTING	InitiatePutOD.ind
		&& consequence = loading-free-of-interaction
			InitiatePutOD.rsp(-){
			ErrorType = object-state-conflict
			}
				OD-LOADING-INTERACTING

S11	OD-LOADING-INTERACTING	PutOD.ind	PutOD.rsp(+){ }	OD-LOADING-INTERACTING

S12	OD-LOADING-INTERACTING	TerminatePutOD.ind
		&& OD is not usable
			TerminatePutOD.rsp(-){
			ErrorType = operational-problem
			}
				OD-LOADING-INTERACTING

S13	OD-LOADING-INTERACTING	TerminatePutOD.ind && OD is okay	TerminatePutOD.rsp(+){ }	OD-READY

6.3.3 Elément ASE Context Management

6.3.3.1 Vue d'ensemble

Les services de gestion de contexte sont utilisés pour établir explicitement une VCR, pour libérer une VCR et pour rejeter des services incorrects. Les données nécessaires pour cela sont stockées dans la description d'objets de l'OD et dans l'objet de la VCR. Des objets de transaction sont fournis pour chaque VCR. Les objets de transaction sont utilisés pour la gestion des primitives de service confirmé.

6.3.3.2 Spécifications des classes Context Management

6.3.3.2.1 Spécification de la classe "VCR List" (Liste des VCR)

6.3.3.2.1.1 Modèle formel de la liste des VCR

La Liste des Relations de Communication Virtuelle (Liste des VCR) contient les descriptions de toutes les VCR d'un appareil. Une VCR se compose d'une partie statique et d'une partie dynamique. Ce paragraphe donne la définition à la partie spécifique FMS de la VCR. Se référer à la Spécification de la Gestion du Réseau pour la définition d'une autre partie de la VCR.

ASE: **ASE Context Management**

CLASSE: **VCR List**

CLASS ID: **—**
CLASSE PARENTE: **TOP**
ATTRIBUTS STATIQUES DE LA VCR:

1.1	(m)	Attribut:	Static VCR ID
1.2	(m)	Attribut:	Max APDU Size Sending
1.3	(m)	Attribut:	Max APDU Size Receiving
1.4	(m)	Attribut:	Features Supported Length
1.5	(m)	Attribut:	Features Supported
1.6	(m)	Attribut:	Max Outstanding Services Sending
1.7	(m)	Attribut:	Max Outstanding Services Receiving
1.8	(m)	Attribut:	VFD ID

ATTRIBUTS DYNAMIQUES DE LA VCR:

2.1	(m)	Attribut:	Dynamic VCR ID
2.2	(m)	Attribut:	Outstanding Services Counter Client
2.3	(m)	Attribut:	Outstanding Services Counter Server
2.4	(m)	Attribut:	VCR State
2.5	(m)	Attribut:	Actual Max Outstanding Services Sending
2.6	(m)	Attribut:	Actual Max Outstanding Services Receiving

SERVICES:

1.	(m)	OpsService:	Initiate
2.	(m)	OpsService:	Abort
3.	(m)	OpsService:	Reject

6.3.3.2.1.2 Attributs

ATTRIBUTS STATIQUES DE LA VCR:

Static VCR ID

Cet attribut désigne l'indice dans le Dictionnaire d'Objets pour la partie statique de cette VCR.

Max APDU Sending

Cet attribut spécifie la longueur maximale de l'unité APDU qui peut être envoyée sur cette VCR.

Max APDU Receiving

Cet attribut spécifie la longueur maximale de l'unité APDU qui peut être reçue sur cette VCR.

Features Supported Length

Cet attribut définit le nombre d'octets présent dans le champ "Features Supported".

Features Supported

Cet attribut identifie les services et fonctionnalités pris en charge sur cette VCR, comme présenté au Tableau 17. Ce paragraphe spécifie l'attribution binaire de la partie fixe et la méthode d'extension des services et des fonctionnalités. L'attribut "Features Supported" est de la longueur spécifiée par l'attribut "Features Supported Length". Il se compose de deux chaînes binaires concaténées de même longueur. La première chaîne binaire identifie les services et les fonctionnalités jouant le rôle de client et la seconde, identifie les services et les fonctionnalités jouant le rôle de serveur. Pour chaque service facultatif, il existe deux bits, l'un dans la première chaîne binaire et l'autre dans la seconde. Pour la fonctionnalité facultative, il existe deux bits, l'un dans la première chaîne binaire, spécifiant que la fonctionnalité facultative est admissible. L'extension de l'attribut peut être effectuée en utilisant des bits non attribués ou en ajoutant un autre octet à la fin de chaque chaîne binaire. Tous les bits d'un octet, qui ne sont pas spécifiés ont la valeur 0.

Tableau 17 – Attribut "Features Supported" de la FMS

Service	Primitive des fonctions du client	bit(n)	Primitive des fonctions du serveur	bit(n)
GetOD (Long form)	.req, .cnf	0	.ind, .rsp	0
UnsolicitedStatus	.req	1	.ind	1
InitiatePutOD	.req, .cnf	2	.ind, .rsp	2
PutOD	.req, .cnf	"	.ind, .rsp	"
TerminatePutOD	.req, .cnf	"	.ind, .rsp	"
InitiateDownloadSequence	.req, .cnf	3	.ind, .rsp	3
DownloadSegment	.ind, .rsp	"	.req, .cnf	"
TerminateDownloadSequence	.ind, .rsp	"	.req, .cnf	"
InitiateUploadSequence	.req, .cnf	4	.ind, .rsp	4
UploadSegment	.req, .cnf	"	.ind, .rsp	"
TerminateUploadSequence	.req, .cnf	"	.ind, .rsp	"
RequestDomainDownload	.req, .cnf	5	.ind, .rsp	5
RequestDomainUpload	.req, .cnf	6	.ind, .rsp	6
CreateProgramInvocation	.req, .cnf	7	.ind, .rsp	7
DeleteProgramInvocation	.req, .cnf	"	.ind, .rsp	"
Start	.req, .cnf	8	.ind, .rsp	8
Stop	.req, .cnf	"	.ind, .rsp	
Resume	.req, .cnf		.ind, .rsp	
Reset	.req, .cnf		.ind, .rsp	"
Kill	.req, .cnf	9	.ind, .rsp	9
Read	.req, .cnf	10	.ind, .rsp	10
Write	.req, .cnf	11	.ind, .rsp	11
ReadWithType	.req, .cnf	12	.ind, .rsp	12
WriteWithType	.req, .cnf	13	.ind, .rsp	13
PhysRead	.req, .cnf	14	.ind, .rsp	14
PhysWrite	.req, .cnf	15	.ind, .rsp	15
InformationReport	.req	16	.ind	16
InformationReportWithType	.req	17	.ind	17
DefineVariableList	.req, .cnf	18	.ind, .rsp	18
DeleteVariableList	.req, .cnf	"	.ind, .rsp	"
EventNotification	.req	19	.ind	19
EventNotificationWithType	.req	20	.ind	20
AcknowledgeEventNotification	.req, .cnf	21	.ind, .rsp	21
AlterEventConditionMonitoring	.req, .cnf	22	.ind, .rsp	22
Addressing with Name	.req	23	.ind	23
Extensions	**Primitive des fonctions du client**	**bit[n]**	**Primitive des fonctions du serveur**	**bit[n]**
GenericInitiateDownloadSequence	.req, .cnf	24	.ind, .rsp	24
GenericDownloadSegment	.req, .cnf	"	.ind, .rsp	"
GenericTerminateDownloadSequence	.req, .cnf	"	.ind, .rsp	"
Explication: [n]: 0 à (23+8x); sans extension ==> x = 0; avec extension ==> x = 1,2, ...				

Les services suivants sont obligatoires pour toutes les VCR afin de les ouvrir et de les fermer:

- Initiate - Abort

Par souci de cohérence, le serveur prend en charge au moins les services de base suivants:

- Identify - Status
- Reject - GetOD

Max Outstanding Services Sending (MaxSCC)

Cet attribut spécifie le nombre maximal de services confirmés en cours compétents dans cette station en tant que côté demandeur de cette VCR.

Max Outstanding Services Receiving (MaxRCC)

Cet attribut spécifie le nombre maximal de services confirmés en attente compétents dans cette station en tant que côté répondeur de cette VCR.

VFD Reference

Cet attribut identifie le VFD qui utilise la VCR.

ATTRIBUTS DYNAMIQUES DE LA VCR:

Outstanding Services Counter Client (OSCS)

Il convient que cet attribut spécifie le nombre de services confirmés actuellement en cours sur cette VCR en tant que demandeur.

Outstanding Services Counter Server (OSCR)

Il convient que cet attribut spécifie le nombre de services en cours actuellement traités sur cette VCR en tant que répondeur.

VCR State

Il convient que cet attribut spécifie l'état de la VCR. Une VCR orientée connexion peut être dans l'un des états suivants:

– CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)
– CONNECTION-ESTABLISHING (CALLING) (ETABLISSEMENT-CONNEXION (APPELANT))
– CONNECTION-ESTABLISHING (CALLED) (ETABLISSEMENT-CONNEXION (APPELE))
– CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

Une VCR sans connexion peut être dans l'un des états suivants:

– CONNECTIONLESS-CLIENT (CLIENT SANS CONNEXION)
– CONNECTIONLESS-SERVER (SERVEUR SANS CONNEXION)

Actual Max Outstanding Services Sending (ActualMaxSCC)

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés dans cette station en tant que côté demandeur de cette VCR. Il convient que sa valeur soit déterminée lors de l'initialisation de cette VCR.

Actual Outstanding Services Receiving (ActualMaxRCC)

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés dans cette station en tant que côté répondeur de cette VCR. Il convient que sa valeur soit déterminée lors de l'initialisation de cette VCR.

6.3.3.2.1.3 Services

Initiate

Ce service facultatif est utilisé pour ouvrir la VCR.

Abort

Ce service facultatif est utilisé pour fermer la VCR.

Reject

Ce service facultatif est utilisé pour rejeter une primitive de service soumise à la VCR.

6.3.3.2.2 Spécification de la classe "Transaction Object" (Objet "Transaction")

6.3.3.2.2.1 Modèle formel

Il convient qu'un Objet "Transaction" soit créé à la réception d'une primitive "indication" de service confirmé. Il convient que l'Objet "Transaction" soit relatif de manière unique à la primitive de service correspondante. Il convient que l'Objet "Transaction" soit supprimé après l'envoi de la primitive "response" correspondante. Il convient que l'Objet "Transaction" soit identifié de manière unique par la combinaison d'un InstanceID et d'un ID de VCR Statique créés selon un protocole.

Il convient que le nombre maximal d'Objets "Transaction" pour chaque VCR du serveur soit spécifié par l'attribut "Max Outstanding Services Receiving" dans l'article de la VCR. Les services suivants agissent sur l'Objet "Transaction":

ASE:		**ASE Context Management**
CLASSE:		**Transaction Object**
CLASS ID:		**—**
CLASSE PARENTE:		**TOP**
ATTRIBUTS:		
1	(m)KeyAttribute:	instanceID & VCR ID
2	(m)Attribut:	Confirmed service indication

6.3.3.2.2.2 Attributs

InstanceID & VCR ID

InstanceID

Il convient que cet attribut identifie l'Objet "Transaction" au sein de la VCR.

VCR ID

Il convient que cet attribut identifie la VCR sur laquelle l'Objet "Transaction" est utilisé.

Confirmed service indication

Identificateur et argument du service en attente.

6.3.3.2.2.3 Services

Initiate

Ce service facultatif est utilisé pour ouvrir la VCR.

Abort

Ce service facultatif est utilisé pour fermer brusquement la VCR.

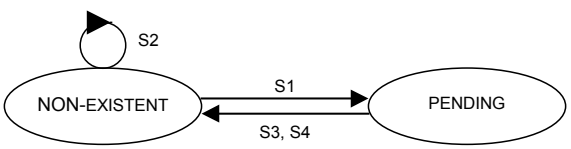
Reject

Ce service est initié de manière interne par l'ASE AP pour indiquer qu'une unité APDU de demande de service confirmé a été reçue avec une erreur de protocole.

6.3.3.2.2.4 Diagramme d'états

6.3.3.2.2.4.1 Description du diagramme d'états

Le diagramme d'états apparaît à la Figure 6.



Légende

Anglais	Français
NON-EXISTENT	NON-EXISTANT
PENDING	EN ATTENTE

Figure 6 – Diagramme d'états de l'objet "Transaction"

NON-EXISTENT (NON EXISTANT)

L'Objet "Transaction" n'existe pas.

PENDING (EN ATTENTE)

Une primitive "indication" de service confirmé a été reçue et la primitive "response" correspondante n'a pas encore été envoyée.

6.3.3.2.2.4.2 Transitions d'états

Le Tableau 18 spécifie les transitions d'états relatives aux objets "transaction".

Tableau 18 – Transitions d'états des objets "transaction"

#	Etat actuel	Evénements	Actions	Etat suivant
S1	NON_EXISTENT	Confirmed service.ind && OSCR < Act MaxRCC	(aucune action entreprise)	PENDING
S2	NON_EXISTENT	Confirmed service.ind && OSCR = Act MaxRCC	(aucune action entreprise)	NON_EXISTENT
S3	PENDING	Confirmed service.rsp	(aucune action entreprise)	NON_EXISTENT
S4	PENDING	Abort.ind	(aucune action entreprise)	NON_EXISTENT

6.3.3.3 Spécification des services ASE de Gestion de Contexte

6.3.3.3.1 Services pris en charge

Ce paragraphe comporte la définition des services uniques à cet élément ASE. Les services définis pour cet ASE sont:

– Initiate
– Abort
– Reject

6.3.3.3.2 Service Initiate

6.3.3.3.2.1 Vue d'ensemble du service

Il convient d'utiliser ce service pour établir une connexion entre deux partenaires de communication et pour échanger des informations relatives aux services pris en charge, aux options prises en charge, à la longueur maximale des unités PDU et à la version actuelle de l'OD.

Lorsque le type d'une VCR est tel que cette PDU Initiate de la FMS n'est pas échangée, la demande est confirmée localement et aucune information relative au partenaire de communication n'est disponible dans la confirmation.

6.3.3.3.2.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 19.

Tableau 19 – Paramètres du service Initiate

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Version OD Calling	M	M (=)		
Profile Number Calling	M	M (=)		
Access Protection Supported Calling	M	M (=)		
Password Calling	M	M (=)		
Access Groups Calling	M	M (=)		
Destination DL-Address	C			
Result(+)			S	S (=)
Version OD Called			M	M (=)
Profile Number Called			M	M (=)
Access Protection Supported Called			M	M (=)
Password Called			M	M (=)
Access Groups Called			M	M (=)
Result(-)			S	S (=)
Error Code			M	M (=)
Max APDU Length Sending Called				C
Max APDU Length Receiving Called				C
Features Supported Called				C

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

Ce paramètre transporte les paramètres de l'invocation du service.

Version OD Calling

Ce paramètre spécifie la version de l'OD du client. Sa valeur est nulle si le client ne contient pas d'OD.

Profile Number Calling

Ce paramètre spécifie la valeur de l'attribut "Profile Number" du client s'il en possède une. S'il n'en a pas, sa valeur est nulle.

Access Protection Supported Calling

Ce paramètre spécifie la valeur de l'attribut "Access Protection Supported" du client s'il en possède une. S'il n'en a pas, sa valeur est nulle.

Password Calling

Ce paramètre spécifie le mot de passe à utiliser pour accéder à tous les objets du serveur sur cette AR. Si l'accès par le biais d'un mot de passe n'est pas utilisé sur cette AR, sa valeur est nulle.

Access Groups Calling

Ce paramètre spécifie l'appartenance du client à des groupes d'accès spécifiques. L'appartenance s'applique pour l'accès à tous les objets du serveur sur cette AR.

Destination DL-Address

Ce paramètre existe uniquement lorsque l'AR correspondante le prend en charge et que le Type de Configuration de l'Adresse Distante est FREE (LIBRE). Il indique l'adresse distante à laquelle il convient d'établir la connexion.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Version OD Called

Ce paramètre spécifie la version de l'OD du serveur.

Profile Number Called

Ce paramètre spécifie la valeur de l'attribut "Profile Number" du serveur.

Access Protection Supported Called

Ce paramètre spécifie la valeur de l'attribut "Access Protection Supported" du serveur s'il en possède une. S'il n'en a pas, sa valeur est nulle.

Password Called

Ce paramètre spécifie le mot de passe à utiliser pour accéder à tous les objets du client sur cette VCR. Si l'accès par le biais d'un mot de passe n'est pas utilisé sur cette VCR, sa valeur est nulle.

Access Groups Called

Ce paramètre spécifie l'appartenance du serveur à des groupes d'accès spécifiques. L'appartenance s'applique pour l'accès à tous les objets du client sur cette VCR.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Error Code

Il convient que ce paramètre fournisse la cause de l'échec telle que spécifiée dans le Tableau 20.

Tableau 20 – Causes d'échec

Cause	Signification
Max APDU Size Insufficient	La longueur maximale des unités APDU n'est pas suffisante pour la communication.
Feature Not Supported	Le service ou l'option demandé(e) n'est pas pris(e) en charge par le serveur.
User Initiate Denied	L'utilisateur refuse d'établir la connexion.
Version Object Definition incompatible	Les versions appelées et appelantes des Définitions d'Objets ne sont pas compatibles.
Password Error	Il existe déjà une VCR établie avec le même mot de passe, ou le mot de passe n'est pas valide.
Profile number incompatible	Le numéro de profil du client n'est pas pris en charge par le serveur.
Other	Cause autre que celles identifiées ci-dessus

Max PDU Length Sending Called

Ce paramètre spécifie la longueur maximale de l'unité APDU à envoyer par le serveur sur cette VCR.

Max PDU Length Receiving Called

Ce paramètre spécifie la longueur maximale des unités APDU à recevoir par le serveur sur cette VCR.

Features Supported Called

Ce paramètre identifie les services et options facultatifs pris en charge par le serveur sur cette VCR.

6.3.3.3.2.3 Procédure de service

La Procédure de Service Confirmé spécifiée à l'Article 4 s'applique à ce service.

6.3.3.3 Service Abort

6.3.3.3.1 Vue d'ensemble du service

Il convient d'utiliser ce service pour libérer une AR existante entre des AP, ou pour mettre fin à l'implication d'un AREP d'un abonné/d'un récepteur dans une AR.

6.3.3.3.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 21.

Tableau 21 – Paramètres du service Abort

Nom de paramètre	Req	Ind
Argument		
VCR	M	M
Locally Generated		M
Abort Identifier	M	M (=)
Reason Code	M	M (=)
Abort Detail	U	U (=)

Argument

Ce paramètre transporte les paramètres de l'invocation de service.

Locally Generated

Il convient que ce paramètre indique si l'interruption a été générée localement ou par le partenaire de communication.

La valeur "false" (faux) n'est pas autorisée si le paramètre "Abort Identifier" possède la valeur "APO ASE" (ASE APO) et si le paramètre "Reason Code" possède la valeur "VCR Error" (Erreur VCR).

Abort Identifier

Il convient que ce paramètre indique où la cause de l'abandon a été détectée. Les valeurs possibles sont:

USER (UTILISATEUR)

APO ASE (ASE APO)

AR ASE (ASE AR)

DLL

Reason Code

Il convient que ce paramètre spécifie la cause de l'abandon.

Si le paramètre "Abort Identifier" possède la valeur "USER", il convient d'appliquer les valeurs énumérées dans le Tableau 22.

Tableau 22 – Causes d'abandon de valeur "USER"

Cause	Signification
Disconnection	La connexion est libérée par l'utilisateur.
Version Object Definition incompatible	Les versions appelées et appelantes des Définitions d'Objets ne sont pas compatibles.
Password Error	Il existe déjà une AR établie avec le même mot de passe, ou le mot de passe n'est pas valide.
Profile Number incompatible	Le numéro de profil du serveur n'est pas pris en charge par le client.
Limited Services Permitted	L'AP est dans le Statut Logique LIMITED-SERVICES-PERMITTED (SERVICES LIMITES AUTORISES).

Si le paramètre "Abort Identifier" possède la valeur "APO ASE" (ASE APO), il convient d'appliquer les valeurs énumérées dans le Tableau 23.

Tableau 23 – Causes d'abandon de valeur "APO ASE" (ASE APO)

Cause	Signification
VCR Error	VCR défaillante
User Error	Primitive de service incorrecte, inconnue ou défectueuse reçue de la part de l'utilisateur
APDU Error	APDU inconnue ou défectueuse reçue
Connection State Conflict AR ASE	Primitive de service ASE AR incorrecte
AR ASE Error	Primitive de service ASE AR inconnue ou défectueuse
APDU Size	La longueur des unités APDU dépasse la longueur maximale des unités APDU
Feature Not Supported	SERVICE-REQ PDU reçue de la part de l'ASE AR et service ou option non pris en charge comme serveur (voir l'attribut "Features Supported" dans la VCR)
InstanceID Error Response	Confirmed service.rsp reçue de la part de l'utilisateur et InstanceID n'existe pas ou CONFIRMED_SERVICE-RSP_PDU reçue de la part de l'ASE AR et InstanceID n'existe pas
Max Services Overflow	CONFIRMED_SERVICE-REQ_PDU reçue de la part de l'ASE AR et Outstanding Services Counter Receiving ≥ Max Outstanding Services Receiving
Connection State Conflict	INITIATE-REQ_PDU reçue de la part de l'ASE AR
Service Error	Le service dans la réponse ne correspond pas au service dans l'indication ou le service dans la confirmation ne correspond pas au service dans la demande.
InstanceID Error Request	CONFIRMED_SERVICE-REQ_PDU reçue de la part de l'ASE AR et InstanceID existe déjà.

Si le paramètre "Abort Identifier" a pour valeur "AR ASE" (ASE AR) ou "DLL" (DLL), il convient que le paramètre "Reason Code" soit appliqué par l'ASE AR.

Abort Detail

Ce paramètre facultatif spécifie les informations supplémentaires relatives à la cause de l'abandon (16 octets maximum). Dans le cas de rapports d'erreur de l'application, la signification est définie dans le profil.

6.3.3.3.3 Procédure de service

La Procédure de Service non Confirmé spécifiée à l'Article 4 s'applique à ce service.

6.3.3.4 Service Reject

6.3.3.4.1 Vue d'ensemble du service

Ce service est utilisé pour informer le point d'extrémité distant qu'une erreur de protocole a été détectée. Il est généré par l'ASE AP si une unité APDU a été reçue avec un code de type de service invalide. Lorsqu'une autre ASE APO détecte une erreur de protocole, elle demande en interne à l'ASE AP de générer une unité PDU de Rejet. Ce service n'est pas utilisé pour indiquer qu'une erreur de service a été détectée par l'utilisateur.

6.3.3.4.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 24.

Tableau 24 – Paramètres du service Reject

Nom de paramètre	Ind
Argument	
VCR	M (=)
Detected Here	M
Original InstanceID	U
Reject APDU Type	M
Reject Code	M

Argument

L'argument contient les paramètres de la demande de service.

Detected Here

Ce paramètre indique si l'erreur a été détectée localement (TRUE) (VRAI). La valeur FALSE (FAUX) n'est autorisée que si le paramètre "Reject PDU Type" a pour valeur "Confirmed-Response-PDU" (PDU de Réponse Confirmée) et si le paramètre "Reject Code" a pour valeur "PDU Size" (Taille de PDU).

Original InstanceID

Ce paramètre spécifie l'InstanceID de l'unité APDU rejetée, s'il en possède un.

Reject APDU Type

Il convient que ce paramètre indique le type de la PDU rejetée. Les valeurs admissibles sont comme suit:

Confirmed-Request-PDU (PDU de Demande Confirmée)

Confirmed-Response-PDU (PDU de Réponse Confirmée)

Unconfirmed-PDU (PDU non confirmée)

Unknown type of PDU (Type de PDU inconnu)

Reject Code

Ce paramètre indique la cause du rejet telle qu'énumérée dans le Tableau 25.

Tableau 25 – Cause de rejet de l'unité APDU

Cause	Signification
Invoke-ID-Exists	Confirmed service.req reçue de la part de l'utilisateur et InstanceID existe déjà
Max-Services-Overflow.	Confirmed service.req reçue de la part de l'utilisateur et OSCS = ActualMaxSCC
Feature-Not-Supported-Connection-Oriented	Service.req reçue de la part de l'utilisateur et service ou option non pris en charge comme client (voir l'attribut "Features Supported" dans la VCR)
Feature-Not-Supported-Connectionless.	Unconfirmed service.req reçue de la part de l'utilisateur et service ou option non pris en charge comme client (voir l'attribut "Features Supported" dans la VCR) ou confirmed service.req reçue de la part de l'utilisateur
APDU-Size	La longueur de l'APDU dépasse la longueur maximale des unités APDU
User-Error-Connectionless	Primitive de service incorrecte ou défectueuse reçue de la part de l'utilisateur
Other	Cause autre que celles identifiées ci-dessus.

6.3.3.3.4.3 Procédure de service

Si un ASE APO reçoit une unité APDU qui contient l'une des erreurs ci-dessus, l'ASE AP délivre une primitive "indication" de rejet.

Si une APO ASE reçoit une primitive "request" confirmée qui contient l'une des erreurs ci-dessus, l'ASE AP délivre une primitive "indication" de rejet.

Si une ASE APO reçoit une primitive "response" confirmée qui contient l'une des erreurs ci-dessus, l'ASE AP élabore une unité APDU de Demande de Rejet et l'envoie sur l'AR concernée.

A la réception de l'unité APDU de Demande de Rejet, l'ASE AP réceptrice délivre une primitive "indication" de Rejet de l'AR à l'utilisateur demandant.

6.3.3.4 Essais de l'élément ASE de gestion de contexte à l'établissement de la connexion

6.3.3.4.1 Essai du contexte dans l'AE

A la réception d'une INITIATE-REQ_PDU, il convient que l'AE appelée vérifie si le contexte du partenaire de communication (Context distant) est compatible avec son propre contexte (contexte local) comme défini pour cette connexion dans cette VCR, comme présenté au Tableau 26. Le contexte local est supposé comme correct. La compatibilité du contexte local avec le contexte distant est définie par le tableau suivant. Dans ce tableau, les champs dépourvus de sens restent vides (par exemple, l'association d'un contexte local "Max APDU Sending" à un contexte distant "Features Supported"). Il convient que ces associations ne soient pas vérifiées.

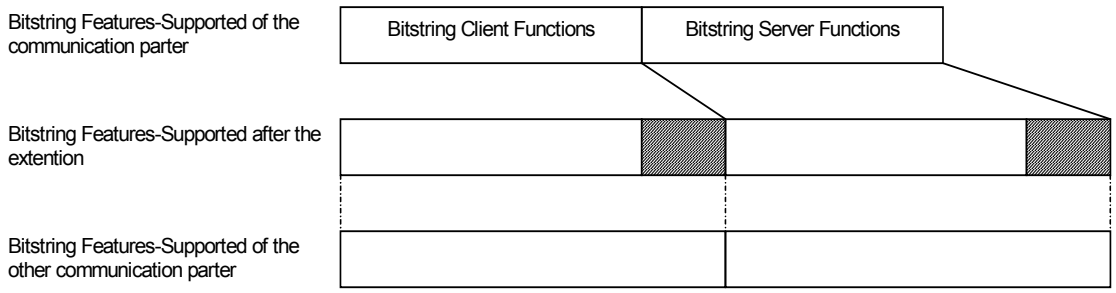
Tableau 26 – Compatibilité du contexte local avec le contexte distant

Contenu distant			Contenu local					
			Max APDU		Feature Supported			
					Req [n]		Ind [n]	
			Emission	Réception	0	1	0	1
Max APDU Sending				≥				
Max APDU Receiving			≤					
Features Supported	.req [n]	0					X	X
	.req [n]	1					-	X
	.ind [n]	0			X	-		
	.ind [n]	1			X	X		

Explication:

.req 0:	La fonctionnalité n'est pas utilisée comme Client
.req 1:	La fonctionnalité est utilisée comme Client
.ind 0:	La fonctionnalité n'est pas prise en charge comme Serveur
.ind 1:	La fonctionnalité est prise en charge comme Serveur
≤ :	valeur locale inférieure ou égale à la valeur distante
≥ :	valeur locale supérieure ou égale à la valeur distante
X :	compatible
– :	non compatible (cas d'erreur)
[n] :	0 à (23+8x); sans extension ==> x= 0; avec extension ==> x= 1,2, ...

La longueur de "Features-Supported" n'est pas limitée. Il peut donc arriver que la longueur de la chaîne binaire de "Features-Supported" soit différente de la longueur de la chaîne binaire de "Features-Supported" des partenaires de communication. Pour vérifier le contexte, il est nécessaire de rapprocher les valeurs des longueurs des deux chaînes binaires, qu'il convient de comparer. La règle consiste à augmenter avec des bits de valeur 0 la chaîne binaire la plus courte jusqu'à ce que les deux chaînes binaires soient égales. Noter que la chaîne binaire de "Features Supported" se compose d'une chaîne binaire pour la primitive des fonctions du client et d'une chaîne binaire pour la primitive des fonctions du serveur. L'extension doit être symétrique entre la chaîne binaire pour la primitive des fonctions du client et la chaîne binaire pour la primitive des fonctions du serveur. Ceci est illustré à la Figure 7.



Légende

Anglais	Français
Bitstring Features-Supported of the communication partner	Chaîne binaire de "Features-Supported" du partenaire de communication
Bitstring Features-Supported after the extension	Chaîne binaire de "Features-Supported" après l'extension
Bitstring Feature—Supported of the other communication partner	Chaîne binaire de "Features-Supported" de l'autre partenaire de communication
Bitstring client functions	Chaîne binaire des fonctions du client
Bitstring server functions	Chaîne binaire des fonctions du serveur

Figure 7 – Essai du contexte de deux attributs "features-supported" avec des chaînes binaires de longueur différente

"MaxRCC" et "MaxSCC" du point d'extrémité initiateur apparaissent dans INITIATE-REQ_PDU. A la réception de cette PDU, il convient que l'AE appelée stocke le plus petit "MaxRCC" dans la PDU et son propre "MaxSCC" à son "ActualMaxSCC". Il convient qu'elle stocke le plus petit "MaxSCC" dans la PDU et son propre "MaxRCC" à son "ActualMaxRCC". Il convient que l'AE appelée retourne l'"ActualMaxRCC" et l'"ActualMaxSCC" résultantes dans les champs "MaxRCC" et "MaxSCC" de INITIATE-RSP_PDU. A la réception de cette PDU, il convient que la FMS initiatrice les stocke respectivement dans ses propres "Act MaxSCC" et "ActualMaxRCC",. Si "ActualMaxRCC" est supérieur à "MaxRCC" ou si "ActualMaxSCC" est supérieur à "MaxSCC", il convient que l'AE initiatrice abandonne cette connexion.

6.3.3.4.2 Essai du contexte dans l'utilisateur

Le contexte suivant doit être contrôlé par l'utilisateur AE.

– Essai du mot de passe. Si l'utilisateur prend en charge l'accès par le biais d'un mot de passe, il convient qu'il vérifie que le mot de passe n'est pas ambigu. Cela signifie que s'il y a un mot de passe, il convient qu'il soit différent des mots de passe de toutes les autres VCR.

– Si le mot de passe n'est pas ambigu et que l'utilisateur est le serveur du service Initiate, il convient qu'il produise une primitive Initiate.rsp avec le paramètre "Result(-)" et avec l'"Error Code" "Password Error" (Erreur de Mot de Passe).

– Si le mot de passe n'est pas ambigu et que l'utilisateur est le client du service Initiate, il convient qu'il libère la connexion avec le service Abort avec le "Reason Code" "Password Error" (Erreur de Mot de Passe).

– Essai de la Version de l'OD. Si l'utilisateur possède un OD distant pour cette connexion, il peut vérifier si le paramètre reçu "Version OD" est compatible avec l'attribut "Version OD" de l'OD distant associé.

– S'ils sont incompatibles et que l'utilisateur est le serveur du service Initiate, il peut générer une primitive Initiate.rsp avec le paramètre "Result(-)" et avec l'"Error Code" "Version OD incompatible".

– S'ils sont incompatibles et que l'utilisateur est le client du service Initiate, il peut libérer la connexion avec le service Abort avec le "Reason Code" "Version OD incompatible".

– Essai du Numéro de Profil. L'utilisateur peut vérifier si le paramètre reçu "Profile Number" est compatible avec l'attribut "Profile Number" du VFD.
– S'ils sont incompatibles et que l'utilisateur est le serveur du service Initiate, il peut générer une primitive Initiate.rsp avec le paramètre "Result(-)" et avec l'"Error Code" "Profile Number incompatible".
– S'ils sont incompatibles et que l'utilisateur est le client du service Initiate, il peut libérer la connexion avec le service Abort avec le "Reason Code" "Profile Number incompatible".

6.3.4 ASE de la Relation entre Applications

6.3.4.1 Vue d'ensemble

L'ASE AR prend en charge trois types d'AR, toutes issues de la classe AREP de premier niveau présentée ci-dessous.

6.3.4.2 Spécification de la classe "AR Endpoint" (Point d'extrémité de la relation entre applications)

6.3.4.2.1 Modèle formel

ASE:			**ASE AR**
CLASSE:			**AR ENDPOINT**
CLASS ID:			
CLASSE PARENTE:			**TOP**
ATTRIBUTS:			
1	(m)	KeyAttribute:	Numeric Identifier
SERVICES:			
1	(m)	OpsService:	AR-Associate
2	(m)	OpsService:	AR-Abort
3	(o)	OpsService:	Confirmed Send
4	(o)	OpsService:	Unconfirmed Send
5	(o)	OpsService:	Compel
6	(o)	OpsService:	Get-Buffered-Message
7	(o)	OpsService:	AR-Status

6.3.4.2.2 Attributs

Numeric Identifier

Identificateur numérique de l'AREP.

6.3.4.2.3 Services

AR-Associate

Ce service est utilisé pour établir une AR.

AR-Abort

Ce service est utilisé pour désactiver brusquement une AR.

Confirmed Send

Ce service est utilisé pour envoyer un service confirmé sur l'AR spécifiée.

Unconfirmed Send

Ce service est utilisé pour envoyer un service non confirmé sur l'AR spécifiée.

Compel

Ce service facultatif est utilisé pour décrire une communication acyclique sur un AREP BNU.

Get-Buffered-Message

Ce service facultatif est utilisé pour extraire les contenus actuels du tampon associé.

AR-Status

Ce service facultatif est utilisé pour rapporter le statut de l'AR.

6.3.4.3 Spécifications des services ASE pour la Relation entre Applications

6.3.4.3.1 Service Unconfirmed send

6.3.4.3.1.1 Vue d'ensemble du service

Ce service est utilisé pour envoyer un service non confirmé sur l'AR spécifiée.

6.3.4.3.1.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 27.

Tableau 27 – Paramètres du service Unconfirmed send

Nom de paramètre	Req	Ind
Argument		
AREP	M	M
Remote DLSAP Address	C	C (=)
Duplicate FAL PDU Body		C
FAL APDU Body	M	M (=)
Local Timeliness		C
Remote Timeliness		C

Argument

L'argument contient les paramètres de la demande de service.

Remote DLSAP Address

Ce paramètre conditionnel est présent si l'attribut "ConfigurationType" de l'AREP associé est "Free" (Libre). S'il est présent, ce paramètre transporte l'adresse DLSAP associée à l'AREP distant. Si l'attribut "ConfigurationType" est "Linked" (Lié), ce paramètre n'est pas présent.

Duplicate FAL PDU Body

Ce paramètre conditionnel est présent si l'attribut "DuplicatePduDetectionSupported" est "True" (Vrai). S'il est présent, il indique si le paramètre "FAL APDU Body" est une réplique du paramètre "FAL APDU Body" qui a été reçu et délivré dans une précédente indication de service.

Ce paramètre prend une valeur "True" (vrai) si les mêmes "FAL APDU Bodies" ont été reçus successivement. Dans le cas contraire, il prend une valeur "False" (Faux).

FAL APDU Body

Ce paramètre spécifie une unité APDU de demande de service non confirmé.

Local Timeliness

Ce paramètre conditionnel, s'il est présent et "True" (Vrai), indique que le paramètre "FAL APDU Body" a rempli les critères d'opportunité de réception définis pour la couche DLL. Si sa valeur est "False" (Faux), cela signifie qu'au moins un critère d'opportunité n'a pas été rempli.

Ce paramètre est présent si la valeur de l'attribut "theSubscriberD1TimelinessClass" est différente de "None" (Aucun) et il n'est pas présent si elle est "None" (Aucun).

Remote Timeliness

Ce paramètre conditionnel, s'il est présent et "True" (Vrai), indique que le paramètre "FAL APDU Body" a rempli les critères d'opportunité de l'Editeur et de la DL émettrice. Si sa valeur est "False" (Faux), cela signifie qu'au moins un critère d'opportunité n'a pas été rempli.

Ce paramètre est présent si la valeur de l'attribut "thePublisherD1TimelinessClass" est différente de "None" (Aucun) et il n'est pas présent si elle est "None" (Aucun).

6.3.4.3.2 Service Confirmed send

6.3.4.3.2.1 Vue d'ensemble du service

Ce service est utilisé pour envoyer un service confirmé sur l'AR spécifiée.

6.3.4.3.2.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 28.

Tableau 28 – Paramètres du service Confirmed send

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
AREP	M	M		
FAL APDU Body	M	M (=)		
Result				
FAL APDU Body			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

FAL APDU Body

Ce paramètre spécifie une unité APDU de demande de service confirmé.

Result(+)
FAL APDU Body

Ce paramètre spécifie une unité APDU de réponse de service confirmé.

6.3.4.3.3 Service AR-abort

6.3.4.3.3.1 Vue d'ensemble du service

Ce service est utilisé pour désactiver brusquement une AR.

6.3.4.3.3.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 29.

Tableau 29 – Paramètres du service AR-Abort

Nom de paramètre	Req	Ind
Argument		
AREP	M	M
Locally Generated		M
Identifier	M	M (=)
Reason Code	M	M (=)
Additional Detail	U	U (=)

Argument

L'argument contient les paramètres de la demande de service.

Locally Generated

Ce paramètre indique si le service Abort a été initié sur l'AREP local ou distant.

Identifier

Ce paramètre indique si le service Abort a été initié sur l'AREP local ou distant.

NOTE Lorsque Abort.request ou Abort.indication est générée par l'AREP, le paramètre "Additional Detail" n'existe pas. Si Abort.indication est générée par l'AREP local, les paramètres "Identifier" et "Reason Code" sont mis en place localement.

Reason Code

Ce paramètre spécifie la cause de l'abandon de l'AR.

Additional Detail

Ce paramètre facultatif spécifie des informations supplémentaires relatives à la libération de l'AR.

6.3.4.3.4 Service Compel

6.3.4.3.4.1 Vue d'ensemble du service

Ce service facultatif est utilisé pour décrire une communication acyclique sur un AREP BNU.

6.3.4.3.4.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 30.

Tableau 30 – Paramètres du service Compel

Nom de paramètre	Req	Cnf
Argument		
AREP	M	
Result		
Status		M

Argument

L'argument contient les paramètres de la demande de service.

Result
Status

Ce paramètre indique le résultat de la demande de service. Les trois codes de statut suivants fournis par la couche DLL sont définis:

success (succès)

failure – inappropriate request (échec – demande inappropriée)

failure – reason unspecified (échec – cause non spécifiée)

6.3.4.3.5 Service Get buffered message

6.3.4.3.5.1 Vue d'ensemble du service

Ce service facultatif est utilisé pour extraire les contenus actuels du tampon associé.

6.3.4.3.5.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 31.

Tableau 31 – Paramètres du service Get buffered message

Nom de paramètre	Req	Cnf
Argument		
AREP	M	
Result(+)		S
Duplicate APDU Body		C
FAL PDU Body		M
Local timeliness		C
Remote timeliness		C
Result (-)		S

Argument

L'argument contient les paramètres de la demande de service.

Result(+)
Duplicate FAL PDU Body

Ce paramètre conditionnel est présent si l'attribut "DuplicatePduDetectionSupported" est "True" (Vrai). S'il est présent, il indique si le paramètre "FAL APDU Body" est une réplique du paramètre "FAL APDU Body" qui a été reçu et délivré dans une précédente indication de service.

Ce paramètre a pour valeur "True" (Vrai) si les mêmes "FAL APDU Bodies" ont été reçus successivement. Dans le cas contraire, il a pour valeur "False" (Faux).

FAL APDU Body

Ce paramètre spécifie une unité APDU de demande de service non confirmé.

Local Timeliness

Ce paramètre conditionnel, s'il est présent et "True" (Vrai), indique que le paramètre "FAL APDU Body" a rempli les critères d'opportunité de réception définis pour la couche DLL. Si sa valeur est "False" (Faux), cela signifie qu'au moins un critère d'opportunité n'a pas été rempli.

Ce paramètre est présent si la valeur de l'attribut "theSubscriberD1TimelinessClass" est différente de "None" (Aucun) et il n'est pas présent si elle est "None" (Aucun).

Remote Timeliness

Ce paramètre conditionnel, s'il est présent et "True" (Vrai), indique que le paramètre "FAL APDU Body" a rempli les critères d'opportunité de l'Editeur et de la DL émettrice. Si sa valeur est "False" (Faux), cela signifie qu'au moins un critère d'opportunité n'a pas été rempli.

Ce paramètre est présent si la valeur de l'attribut "PublisherD1TimelinessClass" est différente de "None" (Aucun) et il n'est pas présent si elle est "None" (Aucun).

Result(-)

Ce paramètre indique que la demande de service a échoué. Lorsque ce paramètre est retourné, cela signifie que le tampon rattaché à l'AREP spécifié ne contient pas d'unités PDU FAS.

6.3.4.3.6 Service AR-status

6.3.4.3.6.1 Vue d'ensemble du service

Ce service facultatif est utilisé pour rapporter le statut de l'AR.

6.3.4.3.6.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 32.

Tableau 32 – Paramètres du service AR-Status

Nom de paramètre	Ind
Argument	
AREP	M
Status code	M

Argument

L'argument contient les paramètres de la demande de service.

Status Code

Ce paramètre spécifie l'événement rapporté de la couche DLL. L'événement suivant est spécifié:

Buffer-Sent: La couche DLL a rapporté que le contenu du tampon vient d'être envoyé.

6.3.5 Eléments ASE des Variables

6.3.5.1 Vue d'ensemble

Dans l'environnement de bus de terrain, les processus d'application contiennent des données que des applications distantes sont capables de lire et écrire. L'élément ASE des variables définit les attributs visibles du réseau des données d'application et fournit un ensemble de services pour lire, écrire, et rapporter leurs valeurs. Des services de gestion de la FAL communs sont utilisés pour créer et supprimer des objets de variable et pour accéder à leurs attributs.

6.3.5.2 Spécifications des classes de Modèle de variables

6.3.5.2.1 Spécification de la classe "Simple variable" (Variable simple)

6.3.5.2.1.1 Modèle formel

L'Objet "Simple Variable" représente une variable unique, simple, caractérisée par un type de données défini. La Description d'Objets de la Variable Simple de l'Objet "Variable Access" est stockée de manière statique dans le Dictionnaire d'Objets (S-OD). Le mapping d'une Variable Simple avec une Variable Simple réelle, qui existe dans le système d'application, est défini par la description d'objets de l'Objet "Simple Variable". Il n'y a qu'un seul type de données de l'ensemble des types de données configurés qui est autorisé dans la Description d'Objets de l'Objet "Simple Variable".

ASE:		**ASE DES VARIABLES**
CLASSE:		**SIMPLE VARIABLE**
CLASS ID:		**7**
CLASSE PARENTE:		**TOP**
ATTRIBUTS:		
1	(m) KeyAttribute:	Numeric Identifier
2	(m) Attribut:	Data type Index
3	(m) Attribut:	Length
4	(m) Attribut:	Password
5	(m) Attribut:	Access Groups
6	(m) Attribut:	Access Rights
7	(m) Attribut:	Local Address
8	(m) Attribut:	Extension
SERVICES:		
1	(o) OpsService:	Read
2	(o) OpsService:	Write
3	(o) OpsService:	Information Report

6.3.5.2.1.2 Attributs

Numeric Identifier

Identifie une instance de cette classe d'objets.

Data type Index

Adresse logique du type de données correspondant dans le Dictionnaire des Types Statique (ST-OD).

Length

Longueur en octets des données.

Password

Cet attribut spécifie le mot de passe pour les Droits d'Accès.

Access Groups

Cet attribut met en relation l'objet avec des Groupes d'Accès spécifiques, comme présenté dans le Tableau 33. L'objet est membre d'un groupe lorsque le bit correspondant est mis en place.

Tableau 33 – Appartenance aux groupes d'accès de "Simple Variable"

Bit	Signification
7	Groupe d'Accès 1
6	Groupe d'Accès 2
5	Groupe d'Accès 3
4	Groupe d'Accès 4
3	Groupe d'Accès 5
2	Groupe d'Accès 6
1	Groupe d'Accès 7
0	Groupe d'Accès 8

Access Rights

Cet attribut spécifie les informations relatives aux Droits d'Accès comme présenté au Tableau 34. Le Droit d'Accès spécifique existe lorsque le bit correspondant est mis en place.

Tableau 34 – Appartenance aux droits d'accès de "Simple Variable"

Bit	Nom	Signification
7	R	Droit de lire le mot de passe enregistré
6	W	Droit d'écrire le mot de passe enregistré
3	Rg	Droit de lire les groupes d'accès
2	Wg	Droit d'écrire les groupes d'accès
15	Ra	Droit de lire tous les partenaires de communication
14	Wa	Droit d'écrire tous les partenaires de communication

Local Address

Cet attribut est une adresse spécifique au système de l'objet réel. Il permet l'adressage interne de l'objet. Si aucun mapping de ce type n'est utilisé, il convient que cet attribut prenne la valeur hexadécimale FFFFFFFF.

Extension

Cet attribut spécifie les informations spécifiques au profil.

6.3.5.2.1.3 Services

Read

Ce service autorise un client à lire la valeur d'une variable.

Write

Ce service autorise un client à écrire la valeur d'une variable.

Information Report

Ce service autorise le VFD à envoyer la valeur d'une variable.

6.3.5.2.2 Spécification de la classe "Array Variable" (Variable de matrice)

6.3.5.2.2.1 Modèle formel

L'Objet "Array" est utilisé pour définir une variable construite dans laquelle tous les éléments ont le même type de données et la même longueur.

La Description d'Objets de la Matrice de l'Objet "Variable Access" est stockée de manière statique dans le Dictionnaire d'Objets (S-OD). Le mapping d'une Variable de Matrice avec une matrice réelle, qui existe dans le système d'application, est défini par la description d'objets

de cet objet. La description d'objets pour les Objets "Array" spécifie le nombre d'éléments de la matrice, le type de données et la longueur des éléments.

L'Objet "Array" peut être l'objet d'un accès complet, ou élément par élément à l'aide de sous-indices. Le Sous-indice 1 accède au premier élément de l'objet.

ASE: **ASE VARIABLE**
CLASS: **ARRAY VARIABLE**
CLASS ID: **8**
CLASSE PARENTE: **SIMPLE VARIABLE**
ATTRIBUTS:
1 (m) Attribut: Number of Elements

6.3.5.2.2.2 Attributs

Number of Elements

Indique le nombre d'éléments contenu dans la Matrice.

6.3.5.2.3 Spécification de la classe "Record variable" (Variable d'Enregistrement)

6.3.5.2.3.1 Modèle formel

L'Objet "Record" est composé d'un ensemble de Variables Simples de différents types de Données.

La Description d'Objets de l'Enregistrement de l'Objet "Variable Access" est stockée dynamiquement dans le Dictionnaire d'Objets (S-OD).

Le mapping d'une Variable d'Enregistrement avec un enregistrement réel, qui existe dans le système d'application, est défini par la description d'objets de cet objet. La description d'objets indique la relation de l'objet avec une Description de la Structure des Types de Données configurée.

L'Objet "Record" peut être l'objet d'un accès complet ou élément par élément. S'il convient d'accéder complètement à l'Objet, il convient que l'indice soit utilisé pour le service. S'il convient d'accéder à un élément de l'objet, il convient que le sous-indice soit utilisé avec l'indice du service. Le Sous-indice 1 accède au premier élément de l'objet.

ASE: **ASE VARIABLE**
CLASSE: **RECORD VARIABLE**
CLASS ID: **9**
CLASSE PARENTE: **TOP**
ATTRIBUTS:
1 (m) KeyAttribute: Numeric Identifier
2 (m) Attribut: Data type Index
3 (m) Attribut: Password
4 (m) Attribut: Access Groups
5 (m) Attribut: Access Rights
6 (m) Attribut: List of Local Address
6.1 (m) Attribut: Local Address
7 (m) Attribut: Extension
SERVICES:
1 (o) OpsService: Read
2 (o) OpsService: Write
3 (o) OpsService: Information Report

6.3.5.2.3.2 Attributs

Numeric Identifier

Identifie une instance de cette classe d'objets.

Data type Index

Adresse logique du type de données correspondant dans le Dictionnaire des Types Statique (ST-OD)

Length

Longueur en octets des données.

Password

Cet attribut spécifie le mot de passe pour les Droits d'Accès.

Access Groups

Cet attribut met en relation l'objet avec des Groupes d'Accès spécifiques, comme présenté dans le Tableau 35. L'objet est membre d'un groupe lorsque le bit correspondant est mis en place.

Tableau 35 – Appartenance aux groupes d'accès de "Array Variable"

Bit	Signification
7	Groupe d'Accès 1
6	Groupe d'Accès 2
5	Groupe d'Accès 3
4	Groupe d'Accès 4
3	Groupe d'Accès 5
2	Groupe d'Accès 6
1	Groupe d'Accès 7
0	Groupe d'Accès 8

Access Rights

Cet attribut spécifie les informations relatives aux Droits d'Accès comme présenté au Tableau 36. Le Droit d'Accès spécifique existe lorsque le bit correspondant est mis en place.

Tableau 36 – Appartenance aux droits d'accès de "Array Variable"

Bit	Nom	Signification
7	R	Droit de Lire le Mot de Passe enregistré
6	W	Droit d'Ecrire le Mot de Passe enregistré
3	Rg	Droit de Lire les Groupes d'Accès
2	Wg	Droit d'Ecrire les Groupes d'Accès
15	Ra	Droit de Lire tous les Partenaires de Communication
14	Wa	Droit d'Ecrire tous les Partenaires de Communication

List of Local Address

Cet attribut est une liste d'adresses, une pour chaque élément de l'enregistrement. L'ordre de la liste est défini par l'ordre des éléments dans l'enregistrement.

Local Address

Cet attribut est une adresse spécifique au système de l'objet réel. Il permet l'adressage interne de l'objet. . Si les objets réels sont stockés de manière séquentielle sans espace, alors seule l'Adresse Locale du premier élément est donnée. Si aucun mapping de ce type n'est utilisé, il convient que cet attribut prenne la valeur hexadécimale FFFFFFFF.

Extension

Cet attribut spécifie les informations spécifiques au profil.

6.3.5.2.3.3 Services

Read

Ce service autorise un client à lire la valeur d'une variable.

Write

Ce service autorise un client à écrire la valeur d'une variable.

Information Report

Ce service autorise le VFD à envoyer la valeur d'une variable.

6.3.5.2.4 Spécification de la classe "Variable list" (Liste de Variables)

6.3.5.2.4.1 Modèle formel

L'Objet "Variable List" se compose d'un ensemble d'Objets "Simple Variable", "Array Variable" et "Record Variable".

La Description d'Objets de la Liste de Variables de l'Objet "Variable Access" est stockée dynamiquement dans le Dictionnaire d'Objets (DV-OD). Cette Description d'Objets contient une liste d'indices des objets de variable contenus issue du S-OD.

Un Objet "Variable List" peut être créé et supprimé avec les services DefineVariableList et DeleteVariableList.

Il convient d'indiquer les Droits d'Accès pour le service DefineVariableList. Les Droits d'Accès aux objets uniques sont vérifiés lors de la création (Define VariableList) d'un Objet "Variable List".

L'Objet "Variable List" n'est créé que si les Droits d'Accès requis ne dépassent pas les Droits d'Accès aux objets uniques.

Si le même Objet "Variable List" existe avec les Droits d'Accès requis, aucun nouvel objet n'est créé mais l'Adresse Logique de l'objet existant est transmise au client. Dans le cas contraire, l'Objet "Variable List" est créé avec les Droits d'Accès requis.

Seul le client autorisé avec les Droits d'Accès corrects peut supprimer un Objet "Variable List" (DeleteVariableList).

Si le Dictionnaire d'Objets est fraîchement chargé, tous les Objets "Variable List" sont supprimés.

ASE:			**ASE VARIABLE**
CLASSE:			**VARIABLE LIST**
CLASS ID:			**10**
CLASSE PARENTE:			**TOP**
ATTRIBUTS:			
1	(m)	KeyAttribute:	Numeric IdentAifier
2	(m)	Attribut:	Number of Elements
3	(m)	Attribut:	Password
4	(m)	Attribut:	Access Groups
5	(m)	Attribut:	Access Rights
6	(c)	Attribut:	Deletable

7 (m) Attribut: List Of Element Index
7.1 (m) Attribut: Index
8 (m) Attribut: Extension

SERVICES:

1 (o) OpsService: Read
2 (o) OpsService: Write
3 (o) OpsService: Information Report
4 (o) OpsService: Define Variable List
5 (o) OpsService: Delete Variable List

6.3.5.2.4.2 Attributs

Numeric Identifier

Identifie une instance de cette classe d'objets.

Number Of Elements

Ce paramètre indique le nombre d'Objets de Variable dans la Liste de Variables.

Password

Cet attribut spécifie le mot de passe pour les Droits d'Accès.

Access Groups

Cet attribut met en relation l'objet avec des Groupes d'Accès spécifiques, comme présenté dans le Tableau 37. L'objet est membre d'un groupe lorsque le bit correspondant est mis en place.

Tableau 37 – Appartenance aux groupes d'accès de "Variable List"

Bit	Signification
7	Groupe d'Accès 1
6	Groupe d'Accès 2
5	Groupe d'Accès 3
4	Groupe d'Accès 4
3	Groupe d'Accès 5
2	Groupe d'Accès 6
1	Groupe d'Accès 7
0	Groupe d'Accès 8

Access Rights

Cet attribut spécifie les informations relatives aux Droits d'Accès comme présenté au Tableau 38. Le Droit d'Accès spécifique existe lorsque le bit correspondant est mis en place.

Tableau 38 – Appartenance aux droits d'accès de "Variable List"

Bit	Nom	Signification
7	R	Droit de Lire le Mot de Passe enregistré
6	W	Droit d'Ecrire le Mot de Passe enregistré
5	D	Droit de Supprimer le Mot de Passe enregistré
3	Rg	Droit de Lire les Groupes d'Accès
2	Wg	Droit d'Ecrire les Groupes d'Accès
1	Dg	Droit de Supprimer les Groupes d'Accès
15	Ra	Droit de Lire tous les Partenaires de Communication
14	Wa	Droit d'Ecrire tous les Partenaires de Communication
13	Da	Droit de Supprimer tous les Partenaires de Communication

Deletable

Cet attribut indique si l'Objet "Variable List" peut être supprimé (true) (vrai) ou non (false) (faux) avec le service DeleteVariableList.

List of Element Index

Cet attribut spécifie les indices des Objets "Variable Access" associés à cet Objet "Variable List".

Extension

Cet attribut spécifie les informations spécifiques au profil.

6.3.5.2.4.3 Services

Read

Ce service autorise un client à lire la valeur d'une liste de variables.

Write

Ce service autorise un client à écrire la valeur d'une liste de variables.

Information Report

Ce service autorise le VFD à envoyer la valeur d'une liste de variables.

Define Variable List

Ce service crée un objet "variable list".

Delete Variable List

Ce service supprime un objet "variable list".

6.3.5.2.5 Spécification de la classe "Variable data type" (Type de données des variables)

6.3.5.2.5.1 Modèle formel

Le type de données définit la syntaxe, la plage des variables et leur présentation à l'intérieur du système de communication. Le type de données d'un objet est indiqué par un attribut correspondant dans la Description d'Objets. Les définitions types ne peuvent pas être effectuées à distance par souci d'efficacité. Elles ont une configuration fixe dans le Dictionnaire des Types Statique (ST-OD).

Cette classe définit des types de données configurables libres et des types de données normalisés. Ils sont stockés en commençant par l'Indice 1 dans le Dictionnaire des Types Statique (ST-OD). Les Types de données normalisés qui ne sont pas utilisés sont marqués comme non configurés dans le ST-OD.

ASE:		**ASE VARIABLE**
CLASSE:		**DATA TYPE**
CLASS ID:		
CLASSE PARENTE:		**TOP**
ATTRIBUTS:		
1	(m) KeyAttribute:	Numeric Identifier
2	(m) Attribut:	Description
2.1	(m) Attribut:	Symbolic Name Length
2.2	(m) Attribut:	Symbolic Name

6.3.5.2.5.2 Attributs

Numeric Identifier

Identifie une instance de cette classe d'objets.

Description

Cet attribut spécifie une description textuelle du type de données. Il se compose des attributs "Symbolic Name Length" et "Symbolic Name".

Symbolic Name Length

Cet attribut spécifie la longueur du symbole. La valeur 0 signifie qu'aucun symbole n'est utilisé.

Symbolic Name

Cet attribut spécifie une chaîne visible de 0 à 32 octets en tant que description symbolique.

6.3.5.2.5.3 Services

Aucun.

6.3.5.2.6 Spécification de la classe "Variable data type structure" (Structure des types de données des variables)

6.3.5.2.6.1 Modèle formel

L'Objet "Data type Structure" caractérise la taille et la structure des enregistrements.

La Description d'Objets de l'Objet "Data type Structure" possède une configuration fixe dans le Dictionnaire des Types Statique. Elle contient une liste d'éléments, composée du Type de données et de la Longueur de l'élément d'enregistrement correspondant. Tout comme les Types de données pour les éléments uniques, seuls les Types de données dans l'ensemble des Types de données configurés du Dictionnaire des Types Statique sont autorisés.

ASE:		**ASE VARIABLE**
CLASSE:		**DATA TYPE STRUCTURE**
CLASS ID:		
CLASSE PARENTE:		**TOP**
ATTRIBUTS:		
1	(m) KeyAttribute:	Numeric Identifier
2	(m) Attribut:	Number Of Elements
3	(m) Attribut:	List Of Elements
3.1	(m) Attribut:	Data type Index
3.2	(m) Attribut:	Length

6.3.5.2.6.2 Attributs

Numeric Identifier

Identifie une instance de cette classe d'objets.

Number Of Elements

Nombre d'éléments du Type de données structuré.

List Of Elements

Cet attribut spécifie une liste des éléments, composée du Type de données et de la Longueur de l'élément d'enregistrement correspondant.

Data type Index

Indice dans le ST-OD du Type de données de l'élément "n".

Length

Longueur de l'élément "n" en octets.

6.3.5.2.6.3 Services

Aucun.

6.3.5.3 Spécification des services ASE des variables

6.3.5.3.1 Services pris en charge

Ce paragraphe comporte la définition des services uniques à cet ASE. Les services définis pour cet ASE sont:

Read

Write

Information Report

Define Variable List

Delete Variable List

6.3.5.3.2 Service Read

6.3.5.3.2.1 Vue d'ensemble du service

Ce service permet à un client de lire la valeur d'un objet de variable ou d'un objet de liste de variables.

6.3.5.3.2.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 39.

Tableau 39 – Paramètres du service Read

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Variable Index	M	M (=)		
Variable Subindex	U	U (=)		
Result (+)			S	S (=)
Data			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

Variable Index

Ce paramètre correspond à l'indice OD de la variable ou de la liste de variables.

Variable Subindex

Ce paramètre facultatif identifie un élément individuel dans une variable de matrice ou une variable d'enregistrement par sa position au sein de la variable, en commençant par 1.

Result(+)
Data

Ce paramètre spécifie la valeur de la variable ou de la liste de variables.

6.3.5.3.3 Service Write

6.3.5.3.3.1 Vue d'ensemble du Service

Ce service permet à un client d'écrire la valeur d'un objet de variable ou d'un objet de liste de variables.

6.3.5.3.3.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 40.

Tableau 40 – Paramètres du service Write

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument				
VCR	M	M		
Variable Index	M	M (=)		
Variable Subindex	U	U (=)		
Data	U	U (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. La méthode par laquelle une primitive "response" est corrélée à sa primitive "indication" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

L'argument contient les paramètres de la demande de service.

Variable Index

Ce paramètre correspond à l'indice OD de la variable ou de la liste de variables.

Variable Subindex

Ce paramètre facultatif identifie un élément individuel dans une variable de matrice ou une variable d'enregistrement par sa position au sein de la variable, en commençant par 1.

Data

Ce paramètre spécifie la valeur de la variable ou de la liste de variables.

6.3.5.3.4 Service Information report

6.3.5.3.4.1 Vue d'ensemble du service

Ce service permet au VFD d'envoyer la valeur d'un objet de variable ou d'un objet de liste de variables.

6.3.5.3.4.2 Primitives du service

Les paramètres de service pour ce service sont présentés dans le Tableau 41.

Tableau 41 – Paramètres du service Information report

Nom de paramètre	Req	Ind
Argument		
VCR	M	M
Destination DL-Address	C	
Source DL-Address		C
Duplicated APDU Body		C
Variable Index	M	M (=)
Variable Subindex	U	U (=)
Data	U	U (=)
On Change	U	U (=)

Argument

Ce paramètre transporte les paramètres de l'invocation du service.

Destination DL-Address

Ce paramètre n'existe que lorsque le type de la VCR correspondante est QUU et que le partenaire distant possède un point d'extrémité "FREE" (LIBRE). Il indique l'adresse distante à laquelle il convient d'envoyer les Données requises.

Source DL-Address

Ce paramètre n'existe que lorsque le type de la VCR correspondante est QUU et possède un point d'extrémité "FREE" (LIBRE). Il indique l'adresse depuis laquelle les données indiquées ont été envoyées.

Duplicated APDU Body

Ce paramètre n'existe que lorsque la sous-couche FAS le fournit pour la VCR BNU et indique si le paramètre de données a été reçu et délivré dans une précédente indication de service.

Variable Index

Ce paramètre correspond à l'indice OD de la variable ou de la liste de variables.

Variable Subindex

Ce paramètre facultatif identifie un élément individuel dans une variable de matrice ou une variable d'enregistrement par sa position au sein de la variable, en commençant par 1.

Data

Ce paramètre spécifie la valeur de la variable ou de la liste de variables.

On Change

Ce paramètre Boolean facultatif indique, lorsqu'il est sur "TRUE" (VRAI), que la valeur de la variable est rapportée car elle a été modifiée, ou parce qu'elle n'a pas été modifiée durant une période spécifiée en externe.