# IEC 61158-5-5

Edition 2.0 2014-08

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside

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

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

IEC 61158-5-5:2014-08(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


**About the IEC**
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

**About IEC publications**
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

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

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 It also gives information on projects, replaced and withdrawn publications.

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

**Electropedia - www.electropedia.org**
The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

**IEC Glossary - std.iec.ch/glossary**
More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

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

---

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

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

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

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

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

**Electropedia - www.electropedia.org**
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 14 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

**Glossaire IEC - std.iec.ch/glossary**
Plus de 55 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

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

# IEC 61158-5-5

Edition 2.0 2014-08

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside

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

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

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE CODE PRIX XH

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1734-4

# CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

# INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

## FOREWORD

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

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 61158-5-5 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:

- Added message padding

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

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

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

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.

**IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.**

# 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 5-5: Application layer service definition – Type 5 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 5 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible service provided by the Type 5 fieldbus application layer in terms of

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

b) the primitive actions and events of the service;

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

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

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

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

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

This standard specifies the structure and services of the Type 2 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

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

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

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

### 1.2 Specifications

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

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

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 5 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 61131-3, *Programmable controllers – Part 3: Programming languages*

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

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 61158-5:2014 (all parts), *Industrial communication networks – Fieldbus specifications – Part 5: Application layer service definition*

IEC 61158-6-5, *Industrial communication networks – Fieldbus specifications – Part 6-5: Application layer protocol specification – Type 5 elements*

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: 1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*[1]

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

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

ANSI/IEEE 754-1985, *Binary Floating-Point Arithmetic*

## 3 Terms and definitions

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

### 3.1 ISO/IEC 7498-1 terms

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

[1] Withdrawn.

f) application-process-invocation

g) application-process-type

h) application-service-element

i) application control service element

### 3.4 ISO/IEC 8824 terms

a) object identifier

b) type

### 3.5 Fieldbus data-link layer terms

For the purposes of this document, the following terms as defined in IEC 61158-3-1 and IEC 61158-4-1 apply.

a) DL-Time

b) DL-Scheduling-policy

c) DLCEP

d) DLC

e) DL-connection-oriented mode

f) DLPDU

g) DLSDU

h) DLSAP

i) fixed tag

j) generic tag

k) link

l) MAC ID

m) network address

n) node address

o) node

p) tag

q) scheduled

r) unscheduled

### 3.6 Fieldbus application layer specific terms and definitions

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

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

**3.6.2**
**active connection control object**
instance of a certain FAL class that abstracts the interconnection facility (as Consumer and Provider) of an automation device

**3.6.3**
**address assignment table**
mapping of the client's internal I/O-Data object storage to the decentralised input and output data objects

**3.6.4**
**allocate**
take a resource from a common area and assign that resource for the exclusive use of a specific entity

**3.6.5**
**application**
function or data structure for which data is consumed or produced

**3.6.6**
**application layer interoperability**
capability of application entities to perform coordinated and cooperative operations using the services of the FAL

**3.6.7**
**application objects**
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

**3.6.8**
**application process**
part of a distributed application on a network, which is located on one device and unambiguously addressed

**3.6.9**
**application process identifier**
component that distinguishes multiple application processes used in a device

**3.6.10**
**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.

**3.6.11**
**application process object class**
a class of application process objects defined in terms of the set of their network-accessible attributes and services

**3.6.12**
**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.

**3.6.13**
**application relationship application service element**
application-service-element that provides the exclusive means for establishing and terminating all application relationships

**3.6.14**
**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.

**3.6.15**
**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.

**3.6.16**
**behaviour**
indication of how an object responds to particular events

**3.6.17**
**bit-no**
designates the number of a bit in a bitstring or an octet

**3.6.18**
**channel**
single physical or logical link of an input or output application object of a server to the process

**3.6.19**
**channel related diagnosis**
information concerning a specific element of an input or output application object, provided for maintenance purposes

EXAMPLE: validity of data

**3.6.20**
**class**
set of objects, all of which represent the same kind of system component

Note 1 to entry: A class is a generalisation of an object; a template for defining variables and methods. All objects in a class are identical in form and behaviour, but usually contain different data in their attributes.

**3.6.21**
**class attributes**
attribute that is shared by all objects within the same class

**3.6.22**
**class code**
unique identifier assigned to each object class

**3.6.23**
**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.

**3.6.24**
**client**
a) object which uses the services of another (server) object to perform a task

b) initiator of a message to which a server reacts

**3.6.25**
**configuration check**
comparison of the expected I/O-Data object structuring of the client with the real I/O-Data object structuring to the server in the start-up phase

**3.6.26**
**configuration data base**
interconnection information maintained by the ACCO ASE

**3.6.27**
**configuration fault**
an unacceptable difference between the expected I/O-Data object structuring and the real I/O-Data object structuring, as detected by the server

**3.6.28**
**configuration identifier**
representation of a portion of I/O Data of a single input- and/or output-module of a server

**3.6.29**
**connection**
logical binding between application objects that may be within the same or different devices

Note 1 to entry: Connections may be either point-to-point or multipoint.

**3.6.30**
**connection channel**
description of a connection between a sink and a source of data items

**3.6.31**
**connection ID**
**CID**
identifier assigned to a transmission that is associated with a particular connection between producers and consumers, providing a name for a specific piece of application information

**3.6.32**
**connection path**
an octet stream that defines the application object to which a connection instance applies

**3.6.33**
**connection point**
buffer which is represented as a subinstance of an Assembly object

**3.6.34**
**consume**
act of receiving data from a producer

**3.6.35**
**consumer**
node or sink that is receiving data from a producer

**3.6.36**
**consuming application**
application that consumes data

**3.6.37**
**control commands**
action invocations transferred from client to server to clear outputs, freeze inputs and/or synchronise outputs

**3.6.38**
**conveyance path**
unidirectional flow of APDUs across an application relationship

**3.6.39**
**cyclic**
repetitive in a regular manner

**3.6.40**
**data consistency**
means for coherent transmission and access of the input- or output-data object between and within client and server

**3.6.41**
**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.

**3.6.42**
**device**
physical hardware connected to the link

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

**3.6.43**
**device profile**
a collection of device dependent information and functionality providing consistency between similar devices of the same device type

**3.6.44**
**diagnosis information**
all data available at the server for maintenance purposes

**3.6.45**
**diagnosis information collection**
system diagnosis information that is assembled at the client side

**3.6.46**
**dynamic AR**
AR that requires the use of the AR establishment procedures to place it into an established state

**3.6.47**
**end node**
producing or consuming node

**3.6.48**
**endpoint**
one of the communicating entities involved in a connection

**3.6.49**
**engineering**
abstract term that characterizes the client application or device responsible for configuring an automation system via interconnecting data items

**3.6.50**
**error**
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

**3.6.51**
**error class**
general grouping for related error definitions and corresponding error codes

**3.6.52**
**error code**
identification of a specific type of error within an error class

**3.6.53**
**event**
an instance of a change of conditions

**3.6.54**
**FAL subnet**
subnetworks composed of one or more data link segments, identified by a subset of the network address

Note 1 to entry: FAL subnets are permitted to contain bridges but not routers.

**3.6.55**
**FIFO variable**
a Variable Object class, composed of a set of homogeneously typed elements, where the first written element is the first element that can be read

Note 1 to entry: On the fieldbus only one, complete element can be transferred as a result of one service invocation.

**3.6.56**
**frame**
denigrated synonym for DLPDU

**3.6.57**
**freeze**
function at the DP-slaves for simultaneous data transfer between the input data object and the process

**3.6.58**
**group**
a) <general> a general term for a collection of objects

Specific uses:

b) <addressing> when describing an address, an address that identifies more than one entity

**3.6.59**
**interface**
a) shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

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

**3.6.60**
**interface definition language**
syntax and semantics of describing service parameters in a formal way

Note 1 to entry: This description is the input for the ORPC model, especially for the ORPC wire protocol.

**3.6.61**
**interface pointer**
key attribute that unambiguously addresses an object interface instance

**3.6.62**
**invocation**
act of using a service or other resource of an application process

Note 1 to entry: Each invocation represents a separate thread of control that may be described by its context. Once the service completes, or use of the resource is released, the invocation ceases to exist. For service invocations, a service that has been initiated but not yet completed is referred to as an outstanding service invocation.

**3.6.63**
**I/O data**
object designated to be transferred cyclically for the purpose of processing

**3.6.64**
**identifier related diagnosis**
information dedicated to modules for maintenance purpose

**3.6.65**
**index**
address of an object within an application process

**3.6.66**
**instance**
the actual physical occurrence of an object within a class that identifies one of many objects within the same object class

EXAMPLE California is an instance of the object class US-state.

Note 1 to entry: The terms object, instance, and object instance are used to refer to a specific instance.

**3.6.67**
**instance attributes**
attribute that is unique to an object instance and not shared by the object class

**3.6.68**
**instantiated**
object that has been created in a device

**3.6.69**
**logical device**
a certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

**3.6.70**
**manufacturer ID**
identification of each product manufacturer by a unique number

**3.6.71**
**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.

**3.6.72**
**member**
piece of an attribute that is structured as an element of an array

**3.6.73**
**message router**
object within a node that distributes messaging requests to appropriate application objects

**3.6.74**
**method**
<object> a synonym for an operational service which is provided by the server ASE and invoked by a client

**3.6.75**
**module**
a) <general> hardware or logical component of a physical device

b) <Type 3> addressable unit inside the DP-slave

**3.6.76**
**multipoint connection**
connection from one node to many

Note 1 to entry: Multipoint connections allow messages from a single producer to be received by many consumer nodes.

**3.6.77**
**network**
a set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

**3.6.78**
**object**
abstract representation of a particular component within a device, usually a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behaviour

**3.6.79**
**object remote procedure call**
model for object oriented or component based remote method invocation

**3.6.80**
**object specific service**
service unique to the object class which defines it

**3.6.81**
**originator**
client responsible for establishing a connection path to the target

**3.6.82**
**peer**
role of an AR endpoint in which it is capable of acting as both client and server

**3.6.83**
**physical device**
automation or other network device

**3.6.84**
**point-to-point connection**
connection that exists between exactly two application objects

**3.6.85**
**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.

**3.6.86**
**pre-established AR endpoint**
AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

**3.6.87**
**process data**
object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing

**3.6.88**
**produce**
act of sending data to be received by a consumer

**3.6.89**
**producer**
node that is responsible for sending data

**3.6.90**
**property**
general term for descriptive information about an object

**3.6.91**
**provider**
source of a data connection

**3.6.92**
**providerID**
an unambiguous identifier within the scope of the ACCO assigned by the provider to recognize the internal data of a configured interconnection source

**3.6.93**
**publisher**
role of an AR endpoint that transmits APDUs onto the fieldbus for consumption by one or more subscribers

Note 1 to entry: A publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated AR.

**3.6.94**
**publishing manager**
role of an AR endpoint in which it issues one or more confirmed service request APDUs to a publisher to request the publisher to publish a specified object

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

**3.6.95**
**pull publisher**
type of publisher that publishes an object in response to a request received from its pull publishing manager

**3.6.96**
**pull publishing manager**
type of publishing manager that requests that a specified object be published in a corresponding response APDU

**3.6.97**
**push publisher**
type of publisher that publishes an object in an unconfirmed service request APDU

**3.6.98**
**push publishing manager**
type of publishing manager that requests that a specified object be published using an unconfirmed service

**3.6.99**
**pull subscriber**
type of subscriber that recognizes received confirmed service response APDUs as published object data

**3.6.100**
**push subscriber**
type of subscriber that recognizes received unconfirmed service request APDUs as published object data

**3.6.101**
**quality code**
additional status information of a data item

**3.6.102**
**resource**
a processing or information capability of a subsystem

**3.6.103**
**route endpoint**
object container containing Variable Objects of a variable class

**3.6.104**
**runtime object model**
objects that exist in a device together with their interfaces and methods that are accessible

**3.6.105**
**server**
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

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

**3.6.106**
**service**
operation or function than an object and/or object class performs upon request from another object and/or object class

**3.6.107**
**slot**
address of a module within a DP-slave

**3.6.108**
**subscriber**
role of an AREP in which it receives APDUs produced by a publisher

**3.6.109**
**sync**
function at the DP-slaves for simultaneous data transfer between the output data object and the process

**3.6.110**
**target**
end-node to which a connection is established

**3.6.111**
**unconnected service**
messaging service which does not rely on the set up of a connection between devices before allowing information exchanges

## 3.7 Abbreviations and symbols

| | |
|---|---|
| ACCO | Active Connection Control Object |
| 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 |
| CID | Connection ID |
| CIM | Computer Integrated Manufacturing |
| CIP | Control and Information Protocol |
| CM_API | Actual Packet Interval |
| CM_RPI | Requested Packet Interval |
| Cnf | Confirmation |
| COR | Connection originator |
| CR | Communication Relationship |
| CREP | Communication Relationship End Point |
| DL- | (as a prefix) data-link- |
| DLC | Data-link Connection |
| DLCEP | Data-link Connection End Point |

| | |
|---|---|
| DLL | Data-link layer |
| DLM | Data-link-management |
| DLSAP | Data-link Service Access Point |
| DLSDU | DL-service-data-unit |
| DNS | Domain Name Service |
| DP | Decentralised Peripherals |
| FAL | Fieldbus Application Layer |
| FIFO | First In First Out |
| HMI | Human-Machine Interface |
| ID | Identifier |
| IDL | Interface Definition Language |
| IEC | International Electrotechnical Commission |
| Ind | Indication |
| IP | Internet Protocol |
| ISO | International Organization for Standardization |
| LDev | Logical Device |
| LME | Layer Management Entity |
| O2T | Originator to target (connection characteristics) |
| O⇒T | Originator to target (connection characteristics) |
| ORPC | Object Remote Procedure Call |
| OSI | Open Systems Interconnect |
| PDev | Physical Device |
| PDU | Protocol Data Unit |
| PL | Physical Layer |
| QoS | Quality of Service |
| QC | Quality Code |
| REP | Route Endpoint |
| Req | Request |
| Rsp | Response |
| RT | Runtime |
| SAP | Service Access Point |
| SCL | Security Level |
| SDU | Service Data Unit |
| SEM | State event matrix |
| SMIB | System Management Information Base |
| SMK | System Management Kernel |
| STD | State transition diagram, used to describe object behaviour |
| S-VFD | Simple Virtual Field Device |
| T2O | Target to originator (connection characteristics) |
| T⇒O | Target to originator (connection characteristics) |
| VAO | Variable Object |

## 3.8 Conventions

### 3.8.1 Overview

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

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

### 3.8.2 Conventions for class definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3.8.3 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 series 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. 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 protocol specification standard but not in an IEC 61158-5 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 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

An overview of the data type ASE and the relationships between data types is provided in IEC 61158-1, 10.1.

## 5.2 Formal definition of data type objects

### 5.2.1 Data type class

#### 5.2.1.1 Template

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

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **DATA TYPE ASE** |
| **CLASS:** | **DATA TYPE** | | |
| **CLASS ID:** | | | 5 (FIXED LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY) |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (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 |

#### 5.2.1.2 Attributes

**Data type Numeric Identifier**

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

**Data type Name**

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

**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.

**ATTRIBUTES:**

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

## 5.3 FAL defined data types

### 5.3.1 Fixed length types

#### 5.3.1.1 Boolean types

##### 5.3.1.1.1 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.

##### 5.3.1.1.2 BOOL

This IEC 61131-3 type is the same as Boolean.

##### 5.3.1.1.3 VT_BOOLEAN

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | VT_BOOLEAN |
| 4 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

This data type expresses a Boolean data type with the values TRUE (-1) and FALSE (0) (see Interger16).

#### 5.3.1.2 Bitstring types

##### 5.3.1.2.1 BitString8

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 22 |
| 2 | Data type Name | = | Bitstring8 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 1 |

This type contains 1 element of type BitString.

#### 5.3.1.2.2 OCTET

This IEC 61131-3 type is the same as Bitstring8.

#### 5.3.1.2.3 BitString16

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 23 |
| 2 | Data type Name | = | Bitstring16 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 2 |

#### 5.3.1.2.4 WORD

This IEC 61131-3 type is the same as Bitstring16.

#### 5.3.1.2.5 BitString32

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 24 |
| 2 | Data type Name | = | Bitstring32 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 4 |

#### 5.3.1.2.6 DWORD

This IEC 61131-3 type is the same as Bitstring32.

#### 5.3.1.2.7 BitString64

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 57 |
| 2 | Data type Name | = | Bitstring64 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 8 |

#### 5.3.1.2.8 LWORD

This IEC 61131-3 type is the same as Bitstring64.

### 5.3.1.3 Currency types

#### 5.3.1.3.1 currency

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | currency |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

This data type defines a signed 64-bit integer in units of 1/10,000 (or 1/100 of a cent). A currency number stored as an 8-octet, two's complement integer, scaled by 10,000 to give a fixed-point number with 15 digits to the left of the decimal point and 4 digits to the right. This representation provides a range of ±922337203685477,5807. This data type is useful for calculations involving money, or for any fixed-point calculation where accuracy is particularly important.

#### 5.3.1.4 Date types

##### 5.3.1.4.1 BinaryDate

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 11 |
| 2 | Data type Name | = | BinaryDate |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 7 |

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

##### 5.3.1.4.2 BinaryDate2000

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 51 |
| 2 | Data type Name | = | BinaryDate2000 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

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

##### 5.3.1.4.3 Date

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 50 |
| 2 | Data type Name | = | Date |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 7 |

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours with the most significant bit indicating Standard Time or Daylight Saving Time. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned8 data type and gives the year. The values 0 … 50 correspond to the years 2000 to 2050, the values 51 … 99 correspond to the years 1951 to 1999.

##### 5.3.1.4.4 DATE

| **CLASS:** | **Data type** |
|---|---|

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | DATE |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

This IEC 61131-3 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. It expresses the date as a number of days, starting from 1972.01.01 (January 1st, 1972), the start of the Coordinated Universal Time (UTC) era, until 2151.06.06 (June 6th, 2151), i.e. a total range of 65536 days.

#### 5.3.1.4.5 date

This data type is the same as Float64.

The data type date has a resolution in the range of one nanosecond. It is valid for dates between 1 January 0100 and 31 December 9999. The value 0,0 has been defined for 30 December 1899, 00:00. The integer part of the value represents the days after 30 December 1899 (for dates before this day, the corresponding value is negative); the fractional part defines the time at that day.

#### 5.3.1.4.6 TimeOfDay

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 12 |
| 2 | Data type Name | = | TimeOfDay |
| 4 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 6 |

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date counting the days from January 1, 1984.

#### 5.3.1.4.7 TimeOfDay with date indication

This data type is the same as the TimeOfDay data type defined above.

#### 5.3.1.4.8 TimeOfDay without date indication

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 52 |
| 2 | Data type Name | = | TimeOfDay without date indication |
| 4 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This data type is composed of one element of an unsigned value and expresses the time of day. The element is an Unsigned32 data type and gives the time after the midnight in milliseconds.

#### 5.3.1.4.9 TIME_OF_DAY

This IEC 61131-3 type is the same as TimeofDay without date indication.

#### 5.3.1.4.10 TimeDifference

**CLASS:** **Data type**

**ATTRIBUTES:**

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

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

#### 5.3.1.4.11 TimeDifference with date indication

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 53 |
| 2 | Data type Name | = | TimeDifference with date indication |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 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 second element is an Unsigned16 data type that provides the difference in days.

#### 5.3.1.4.12 TimeDifference without date indication

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 54 |
| 2 | Data type Name | = | TimeDifference without date indication |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This data type is composed of one element of an unsigned value that express the difference in time. The element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds.

#### 5.3.1.4.13 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.

#### 5.3.1.4.14 UniversalTime

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 16 |
| 2 | Data type Name | = | UniversalTime |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 12 |

This simple type is composed of twelve elements of type VisibleString. (YYMMDDHHMMSS). It is the same as that defined in ISO/IEC 8824, except that the local time differential is not supported.

#### 5.3.1.4.15 FieldbusTime

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 17 |
| 2 | Data type Name | = | FieldbusTime |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 7 |

This data type is defined in this standard as DL-Time.

### 5.3.1.5 Enumerated types

#### 5.3.1.5.1 PERSISTDEF

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | PERSISTDEF |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

The allowed values are shown in Table 1.

**Table 1 – PERSISTDEF**

| Value |
|---|
| CBAVolatile |
| CBAPendingPersistent |
| CBAPersistent |

#### 5.3.1.5.2 VARTYPE

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | VARTYPE |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

The VARTYPE specifies the data type that governs the interpretation of the data. The allowed values are shown in Table 2.

**Table 2 – VARTYPE**

| Value | Data type |
|---|---|
| VT_EMPTY | no value |
| VT_NULL | null value (no valid data) |
| VT_BOOL | VT_BOOLEAN |
| VT_I1 | char |
| VT_I2 | short |
| VT_I4 | long |
| VT_UI1 | unsigned char |
| VT_UI2 | unsigned short |
| VT_UI4 | unsigned long |
| VT_R4 | float |

| Value | Data type |
|---|---|
| VT_R8 | double |
| VT_CY | Currency |
| VT_DATE | date |
| VT_BSTR | address of a BSTR |
| VT_SAFEARRAY_BOOL | address of a SAFEARRAY (VT_BOOLEAN) |
| VT_SAFEARRAY_I1 | address of a SAFEARRAY (char) |
| VT_SAFEARRAY_I2 | address of a SAFEARRAY (short) |
| VT_SAFEARRAY_I4 | address of a SAFEARRAY (long) |
| VT_SAFEARRAY_UI1 | address of a SAFEARRAY (unsigned char) specified. |
| VT_SAFEARRAY_UI2 | address of a SAFEARRAY (unsigned short) specified. |
| VT_SAFEARRAY_UI4 | address of a SAFEARRAY (unsigned long) specified. |
| VT_SAFEARRAY_R4 | address of a SAFEARRAY (float) |
| VT_SAFEARRAY_R8 | address of a SAFEARRAY (double) |
| VT_SAFEARRAY_CY | address of a SAFEARRAY (Currency) |
| VT_SAFEARRAY_DATE | address of a SAFEARRAY (date) |
| VT_SAFEARRAY_BSTR | address of a SAFEARRAY (BSTR) |
| VT_DISPATCH | Interface Pointer to an IDispatch interface |
| VT_UNKNOWN | Interface Pointer to an IUnknown interface |
| VT_USERDEFINED | address of an userdefined struct |
| VT_ERROR | A HRESULT is specified. |
| All defined structured data types should have the type code VT_USERDEFINED. | |

#### 5.3.1.5.3 ITEMQUALITYDEF

**CLASS: Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | ITEMQUALITYDEF |
| 3 | Format = | FIXED LENGTH |
| 4.1 | Octet Length = | 1 |

This data type contains the status information of the related data. It consists of three portions: Quality, Substatus and Limits. There are four states of quality (Bad – the value is not useful; Uncertain – the quality of the value is less than normal, but the value may still be useful; Good (Non Cascade) – the quality of the value is good, possible alarm conditions may be indicated by the substatus; Good (Cascade) – the value may be used in control), a set of sub-status values for each quality, and four states of the limits (OK – the value is free to move; low limited (LL) – the value has acceded its low limits, high limited (HL) – the value has acceded its high limits; constant (C) – the value cannot move, no matter what the process does). The allowed values are shown in Table 3.

**Table 3 – ITEMQUALITYDEF**

| Quality | Value | Description |
|---|---|---|
| Bad | BadNonSpecific | There is no specific reason why the value is bad. Used for propagation. |
| | BadNonSpecificLL | and low limited |
| | BadNonSpecificHL | and high limited |
| | BadNonSpecificC | and constant |

| Quality | Value | Description |
|---|---|---|
| | BadConfigurationError | Set if the value is not useful because there is some other problem with the block, depending on what a specific producer can detect. |
| | BadConfigurationErrorLL | and low limited |
| | BadConfigurationErrorHL | and high limited |
| | BadConfigurationErrorC | and constant |
| | BadNotConnected | Set if this input is required to be connected and is not connected. |
| | BadNotConnectedLL | and low limited |
| | BadNotConnectedHL | and high limited |
| | BadNotConnectedC | and constant |
| | BadDeviceFailure | Set if the source of the value is affected by a device failure. |
| | BadDeviceFailureLL | and low limited |
| | BadDeviceFailureHL | and high limited |
| | BadDeviceFailureC | and constant |
| | BadSensorFailure | Set if the device can determine this condition. The limits define which direction has been exceeded. |
| | BadSensorFailureLL | and low limited |
| | BadSensorFailureHL | and high limited |
| | BadSensorFailureC | and constant |
| | BadLastKnownValue | Set if this value had been set by communication, which has now failed. |
| | BadLastKnownValueLL | and low limited |
| | BadLastKnownValueHL | and high limited |
| | BadLastKnownValueC | and constant |
| | BadCommFailure | Set if there has never been any communication with this value since it was last Out of Service. |
| | BadCommFailureLL | and low limited |
| | BadCommFailureHL | and high limited |
| | BadCommFailureC | and constant |
| | BadOutOfService | The value is not reliable because the block is not being evaluated, and may be under construction by a configuration tool. It is set if the block mode is O/S. |
| | BadOutOfServiceLL | and low limited |
| | BadOutOfServiceHL | and high limited |
| | BadOutOfServiceC | and constant |
| Uncertain | UncertainNonSpecific | There is no specific reason why the value is uncertain. Used for propagation. |
| | UncertainNonSpecificLL | and low limited |
| | UncertainNonSpecificHL | and high limited |
| | UncertainNonSpecificC | and constant |
| | UncertainLastUsableValue | Whatever was writing this value has stopped doing so. This is used for fail safe handling. |
| | UncertainLastUsableValueLL | and low limited |
| | UncertainLastUsableValueHL | and high limited |
| | UncertainLastUsableValueC | and constant |
| | UncertainSubstituteSet | Predefined value is used instead of the calculated one. This is used for fail safe handling. |

| Quality | Value | Description |
|---|---|---|
| | UncertainSubstituteSetLL | and low limited |
| | UncertainSubstituteSetHL | and high limited |
| | UncertainSubstituteSetC | and constant |
| | UncertainInitialValue | Value of volatile parameters during and after the reset of the device or a parameter. |
| | UncertainInitialValueLL | and low limited |
| | UncertainInitialValueHL | and high limited |
| | UncertainInitialValueC | and constant |
| | UncertainSensorNotAccurate | Set if the value is at one of the sensor limits. The limits define which direction has been exceeded. Also set if the device can determine that the sensor has reduced accuracy (e.g. degraded analyzer), in which case no limits are set. |
| | UncertainSensorNotAccurateLL | and low limited |
| | UncertainSensorNotAccurateHL | and high limited |
| | UncertainSensorNotAccurateC | and constant |
| | UncertainEngineeringUnitsExceeded | Set if the value lies outside of the range of values defined for this parameter. The limits define which direction has been exceeded. |
| | UncertainEngineeringUnitsExceededLL | and low limited |
| | UncertainEngineeringUnitsExceededHL | and high limited |
| | UncertainEngineeringUnitsExceededC | and constant |
| | UncertainSubNormal | Set if a value derived from multiple values has less than the required number of Good sources. |
| | UncertainSubNormalLL | and low limited |
| | UncertainSubNormalHL | and high limited |
| | UncertainSubNormalC | and constant |
| | UncertainConfigurationError | Set if there is some inconsistency regarding the parameterization or configuration, depending on what a specific producer can detect. |
| | UncertainConfigurationErrorLL | and low limited |
| | UncertainConfigurationErrorHL | and high limited |
| | UncertainConfigurationErrorC | and constant |
| | UncertainSimulatedValue | Set when the process value is written by the operator while the block is in manual mode. |
| | UncertainSimulatedValueLL | and low limited |
| | UncertainSimulatedValueHL | and high limited |
| | UncertainSimulatedValueC | and constant |
| | UncertainSensorCalibration | Set during the active calibration process together with the current measured value. |
| | UncertainSensorCalibrationLL | and low limited |
| | UncertainSensorCalibrationHL | and high limited |
| | UncertainSensorCalibrationC | and constant |
| Good | GoodNonCascOk | No error or special condition is associated with this value. |
| Non Cascade | GoodNonCascOkC | and constant |
| | GoodNonCascActiveUpdateEvent | Set if the value is good and the block has an active Update Event. |
| | GoodNonCascActiveUpdateEventLL | and low limited |
| | GoodNonCascActiveUpdateEventHL | and high limited |

| Quality | Value | Description |
|---|---|---|
| | GoodNonCascActiveUpdateEventC | and constant |
| | GoodNonCascActiveAdvisoryAlarm | Set if the value is good and the block has an active Alarm with a priority less than 8. |
| | GoodNonCascActiveAdvisoryAlarmLL | and low limited |
| | GoodNonCascActiveAdvisoryAlarmHL | and high limited |
| | GoodNonCascActiveAdvisoryAlarmC | and constant |
| | GoodNonCascActiveCriticalAlarm | Set if the value is good and the block has an active Alarm with a priority greater than or equal to 8. |
| | GoodNonCascActiveCriticalAlarmLL | and low limited |
| | GoodNonCascActiveCriticalAlarmHL | and high limited |
| | GoodNonCascActiveCriticalAlarmC | and constant |
| | GoodNonCascUnackUpdateEvent | Set if the value is good and the block has an unacknowledged Update Event. |
| | GoodNonCascUnackUpdateEventLL | and low limited |
| | GoodNonCascUnackUpdateEventHL | and high limited |
| | GoodNonCascUnackUpdateEventC | and constant |
| | GoodNonCascUnackAdvisoryAlarm | Set if the value is good and the block has an unacknowledged Alarm with a priority less than 8. |
| | GoodNonCascUnackAdvisoryAlarmLL | and low limited |
| | GoodNonCascUnackAdvisoryAlarmHL | and high limited |
| | GoodNonCascUnackAdvisoryAlarmC | and constant |
| | GoodNonCascUnackCriticalAlarm | Set if the value is good and the block has an unacknowledged Alarm with a priority greater than or equal to 8. |
| | GoodNonCascUnackCriticalAlarmLL | and low limited |
| | GoodNonCascUnackCriticalAlarmHL | and high limited |
| | GoodNonCascUnackCriticalAlarmC | and constant |
| | GoodNonCascInitialFailSafe | This value is from a block that wants its following output block (e.g. AO) to go to Fail Safe. |
| | GoodNonCascInitialFailSafeLL | and low limited |
| | GoodNonCascInitialFailSafeHL | and high limited |
| | GoodNonCascInitialFailSafeC | and constant |
| | GoodNonCascMaintenanceRequired | The device works still without failure but service support will be necessary soon. This may be detected e.g. by a Transducer Block of a value of pH meter. |
| | GoodNonCascMaintenanceRequiredLL | and low limited |
| | GoodNonCascMaintenanceRequiredHL | and high limited |
| | GoodNonCascMaintenanceRequiredC | and constant |
| Good | GoodCascOk | No error or special condition is associated with this value. |
| Cascade | GoodCascOkC | and constant |
| | GoodCascInitializationAcknowledge | The value is an initialized value from a source (cascade input, remote-cascade in, and remote-output in parameters). |
| | GoodCascInitializationAcknowledgeLL | and low limited |
| | GoodCascInitializationAcknowledgeHL | and high limited |
| | GoodCascInitializationAcknowledgeC | and constant |
| | GoodCascInitializationRequest | The value is an initialization value for a source (back calculation input parameter), because the lower loop is broken or the mode is wrong. |

| Quality | Value | Description |
|---|---|---|
| | GoodCascInitializationRequestLL | and low limited |
| | GoodCascInitializationRequestHL | and high limited |
| | GoodCascInitializationRequestC | and constant |
| | GoodCascNotInvited | The value is from a block which does not have a target mode that would use this input. This covers all cases other than Fail Safe Active, Local Override, and Not Selected. The target mode can be the next permitted mode of higher priority in the case of shedding a supervisory computer. |
| | GoodCascNotInvitedLL | and low limited |
| | GoodCascNotInvitedHL | and high limited |
| | GoodCascNotInvitedC | and constant |
| | GoodCascDoNotSelect | The value is from a block which should not be selected, due to conditions in or above the block. |
| | GoodCascDoNotSelectLL | and low limited |
| | GoodCascDoNotSelectHL | and high limited |
| | GoodCascDoNotSelectC | and constant |
| | GoodCascLocalOverride | The value is from a block that has been locked out by a local key switch or is a Complex AO/DO with interlock logic active. The failure of normal control should be propagated to a PID block for alarm and display purposes. This also implies Not Invited. |
| | GoodCascLocalOverrideLL | and low limited |
| | GoodCascLocalOverrideHL | and high limited |
| | GoodCascLocalOverrideC | and constant |
| | GoodCascInitiateFailSafe | The value is from a block that wants its downstream output block (e.g. AO) to go to Fail Safe. This is determined by a block option to initiate Fail Safe if the status of the primary input and/or cascade input goes Bad. |
| | GoodCascInitiateFailSafeLL | and low limited |
| | GoodCascInitiateFailSafeHL | and high limited |
| | GoodCascInitiateFailSafeC | and constant |

#### 5.3.1.5.4 STATEDEF

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name | = STATEDEF |
| 3 | Format | = FIXED LENGTH |
| 4.1 | Octet Length | = 2 |

The allowed values are shown in Table 4.

**Table 4 – STATEDEF**

| Value |
|---|
| CBANonExistent |
| CBAInitializing |
| CBAReady |
| CBAOperating |
| CBADefect |

#### 5.3.1.5.5 GROUPERRORDEF

**CLASS: Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | GROUPERRORDEF |
| 3 | Format = | FIXED LENGTH |
| 4.1 | Octet Length = | 2 |

The allowed values are shown in Table 5.

**Table 5 – GROUPERRORDEF**

| Value |
|---|
| CBANonAccessible |
| CBAOkay |
| CBAProblem |
| CBAUnknown |

#### 5.3.1.5.6 ACCESSRIGHTSDEF

**CLASS: Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | ACCESSRIGHTSDEF |
| 3 | Format = | FIXED LENGTH |
| 4.1 | Octet Length = | 2 |

The allowed values are shown in Table 6.

**Table 6 – ACCESSRIGHTSDEF**

| Value |
|---|
| CBANoAccess |
| CBAReadAccess |
| CBAWriteAccess |
| CBAFullAccess |

### 5.3.1.6 Handle types

#### 5.3.1.6.1 HRESULT

**CLASS: Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | HRESULT |
| 3 | Format = | FIXED LENGTH |
| 4.1 | Octet Length = | 4 |

The allowed values are shown in Table 7. The defined HRESULT values may be extended by user specific values. If necessary the coding scheme as defined in IEC 61158-6-5 should be applied. In particular, the severity code to indicate success or error should be used.

**Table 7 – HRESULT**

| Value | Description |
|---|---|
| CBA_E_MALFORMED | The identifier is malformed (too long, unallowed characters, no separation characters, syntactically invalid) |
| CBA_E_UNKNOWNOBJECT | The object specified in the item identifier is not known. |
| CBA_E_UNKNOWNMEMBER | The member specified in the item identifier is not known. |
| CBA_E_TYPEMISMATCH | The type specified does not match the expected type. |
| CBA_E_INVALIDENUMVALUE | The enumeration value is invalid. |

| Value | Description |
|---|---|
| CBA_E_INVALIDID | The ConsumerID or ProviderID is not valid. |
| CBA_E_INVALIDEPSILON | The Epsilon type or value is not valid. |
| CBA_E_INVALIDSUBSTITUTE | The Substitute type or value is not valid. |
| CBA_E_INVALIDCONNECTION | It is not allowed to specify a connection from an identifier to itself. |
| CBA_E_INVALIDCOOKIE | The Cookie value is not valid. |
| CBA_E_TIMEVALUEUNSUPPORTED | The time value is unsupported. |
| CBA_E_QOSTYPEUNSUPPORTED | The QoS type is unsupported. |
| CBA_E_QOSVALUEUNSUPPORTED | The QoS value is unsupported. |
| CBA_E_PERSISTRUNNING | While the Save service is running, no changes are allowed to the configuration data base (try again in a few seconds). |
| CBA_E_INUSE | The destination specified in the item identifier is already connected to some other provider. |
| CBA_E_NOTAPPLICABLE | The operation is currently not applicable. |
| CBA_E_NONACCESSIBLE | The item is not accessible. |
| CBA_E_DEFECT | Hardware defect detected, replacement needed. |
| CBA_S_PERSISTPENDING | The value requested is currently not persistent. |
| CBA_S_ESTABLISHING | The connection is not yet established. |
| CBA_S_NOCONNECTION | There is no connection from the provider to this specific consumer. |
| CBA_S_VALUEBUFFERED | The value was only buffered and has no immediate effect on the process (e.g. device is in state CBAReady). |
| CBA_S_VALUEUNCERTAIN | The value requested is currently uncertain. |
| E_OUTOFMEMORY | Insufficient memory to complete the call. |
| E_INVALIDARG | One or more arguments are invalid. |
| E_NOTIMPL | The service is not implemented. |
| E_FAIL | Unspecified error. |
| E_NOINTERFACE | No such interface supported. |
| RPC_S_PROCNUM_OUT_OF_RANGE | The procedure number is out of range. |
| RPC_E_INVALID_OXID | The object exporter was not found. |
| RPC_E_INVALID_OID | The specified object was not found or recognized. |
| RPC_E_INVALID_SET | The object exporter set was not found. |
| RPC_E_INVALID_OBJECT | The requested object does not exist. |
| RPC_E_VERSION_MISMATCH | The ORPC version on the client and server machine does not match. |
| DISP_E_BADINDEX | Invalid index. |
| DISP_E_BADPARAMCOUNT | The number of elements provided to DISPPARAMS is different from the number of arguments accepted by the method or property. |
| DISP_E_BADVARTYPE | One of the arguments is not a valid variant type. |
| DISP_E_EXCEPTION | The application needs to raise an exception. In this case, the structure passed in pExcepInfo should be filled in. |
| DISP_E_MEMBERNOTFOUND | The requested member does not exist, or the call to Invoke tried to set the value of a read-only property. |
| DISP_E_NONAMEDARGS | This implementation of IDispatch does not support named arguments |
| DISP_E_OVERFLOW | One of the arguments could not be coerced to the specified type. |
| DISP_E_PARAMNOTFOUND | One of the parameter DISPIDs does not correspond to a parameter on the method. In this case, puArgErr should be set to the first argument that contains the error. |
| DISP_E_TYPEMISMATCH | One or more of the arguments could not be coerced. The index within rgvarg of the first parameter with the incorrect type is returned in the puArgErr parameter. |

| Value | Description |
|---|---|
| DISP_E_UNKNOWNINTERFACE | The interface identifier passed in riid is not IID_NULL. |
| DISP_E_UNKNOWNLCID | Unknown language. |
| DISP_E_UNKNOWNNAME | Unknown name. |
| DISP_E_PARAMNOTOPTIONAL | A required parameter was omitted. |
| TYPE_E_ELEMENTNOTFOUND | Element not found. |
| S_OK | Success, "everything worked". |
| S_FALSE | Success but with additional results, "function worked and the result is false". |

#### 5.3.1.7 Numeric types

##### 5.3.1.7.1 BCD

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 35 |
| 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. In this type, the least significant four bits are used to express a BCD value which is between zero and nine inclusive. The most significant four bits are unused.

##### 5.3.1.7.2 Floating Point types

###### 5.3.1.7.2.1 Float32

**CLASS:** **Data type**

**ATTRIBUTES:**

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

This type has a length of four octets. The format for float32 is that defined by ANSI/IEEE 754 as single precision.

###### 5.3.1.7.2.2 Floating point

This data type is the same as Float32.

###### 5.3.1.7.2.3 REAL

This IEC 61131-3 type is the same as Float32.

###### 5.3.1.7.2.4 float

This data type is the same as Float32.

#### 5.3.1.7.2.5 Float64

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 15 |
| 2 | Data type Name | = | Float64 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

This type has a length of eight octets. The format for float64 is that defined by ANSI/IEEE 754 as double precision.

#### 5.3.1.7.2.6 LREAL

This IEC 61131-3 type is the same as Float64.

#### 5.3.1.7.2.7 double

This data type is the same as Float64.

### 5.3.1.7.3 Integer types

#### 5.3.1.7.3.1 Integer8

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 2 |
| 2 | Data type Name | = | Integer8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 1 |

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

#### 5.3.1.7.3.2 SINT

This IEC 61131-3 type is the same as Integer8.

#### 5.3.1.7.3.3 char

This data type is the same as Integer8.

#### 5.3.1.7.3.4 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.

#### 5.3.1.7.3.5 INT

This IEC 61131-3 type is the same as Integer16.

#### 5.3.1.7.3.6 short

This data type is the same as Integer16.

#### 5.3.1.7.3.7 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.

#### 5.3.1.7.3.8 DINT

This IEC 61131-3 type is the same as Integer32.

#### 5.3.1.7.3.9 long

This data type is the same as Integer32.

#### 5.3.1.7.3.10 Integer64

| CLASS: | | | Data type |
|---|---|---|---|
| ATTRIBUTES: | | | |
| 1 | Data type Numeric Identifier | = | 55 |
| 2 | Data type Name | = | Integer64 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

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

#### 5.3.1.7.3.11 LINT

This IEC 61131-3 type is the same as Integer64.

### 5.3.1.7.4 Unsigned types

#### 5.3.1.7.4.1 Unsigned8

| CLASS: | | | Data type |
|---|---|---|---|
| ATTRIBUTES: | | | |
| 1 | Data type Numeric Identifier | = | 5 |
| 2 | Data type Name | = | Unsigned8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 1 |

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

#### 5.3.1.7.4.2 USINT

This IEC 61131-3 type is the same as Unsigned8.

#### 5.3.1.7.4.3 unsigned char

This data type is the same as Unsigned8.

#### 5.3.1.7.4.4 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.

#### 5.3.1.7.4.5 UINT

This IEC 61131-3 type is the same as Unsigned16.

#### 5.3.1.7.4.6 unsigned short

This data type is the same as Unsigned16.

#### 5.3.1.7.4.7 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.

#### 5.3.1.7.4.8 UDINT

This IEC 61131-3 type is the same as Unsigned32.

#### 5.3.1.7.4.9 unsigned long

This data type is the same as Unsigned32.

#### 5.3.1.7.4.10 Unsigned64

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 56 |
| 2 | Data type Name | = | Unsigned64 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

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 eight octets.

#### 5.3.1.7.4.11 ULINT

This IEC 61131-3 type is the same as Unsigned64.

#### 5.3.1.8 OctetString character types

##### 5.3.1.8.1 OctetString1

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 30 |
| 2 | Data type Name | = | OctetString1 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 1 |

This type has a length of one octet.

##### 5.3.1.8.2 OctetString2

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 31 |
| 2 | Data type Name | = | OctetString2 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

This type has a length of two octets.

##### 5.3.1.8.3 OctetString4

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 32 |
| 2 | Data type Name | = | OctetString4 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This type has a length of four octets.

##### 5.3.1.8.4 OctetString8

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 33 |
| 2 | Data type Name | = | OctetString8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

This octet string type has a length of eight octets.

##### 5.3.1.8.5 OctetString16

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 34 |
| 2 | Data type Name | = | OctetString16 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 16 |

This type has a length of 16 octets.

##### 5.3.1.8.6 UUID

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | UUID |
| 3 | Format | = STRUCTURE |
| 5.1 | Number of Fields | = 4 |
| 5.2.1 | Field Name | = Data1 |
| 5.2.2 | Field Data type | = unsigned long |
| 5.2.3 | Field Name | = Data2 |
| 5.2.4 | Field Data type | = unsigned short |
| 5.2.5 | Field Name | = Data3 |
| 5.2.6 | Field Data type | = unsigned short |
| 5.2.7 | Field Name | = Data4 |
| 5.2.8.1 | Format | = ARRAY |
| 5.2.8.4.1 | Number of Array Elements | = 8 |
| 5.2.8.4.2 | Array Element Data type | = unsigned char |

This data type defines a 16 octet fixed length data type. The semantic is specified by the used ORPC model and beyond the scope of this standard.

This data type is structured as follows.

**Data1**
This field contains the first eight hexadecimal digits of the UUID.

**Data2**
This field contains the first group of four hexadecimal digits of the UUID.

**Data3**
This field contains the second group of four hexadecimal digits of the UUID.

**Data4**
This field contains an array of eight elements. The first two elements contain the third group of four hexadecimal digits of the UUID. the remaining six elements contain the final 12 hexadecimal digits of the UUID.

Predefined values for TYPE 10 items are shown in Table 8.

**Table 8 – UUID**

| Value | Description |
|---|---|
| UUID_NULL | |
| UUID_IUnknown | Identifies the IUnknown interface uniquely. |
| UUID_IDispatch | Identifies the IDispatch interface uniquely. |
| UUID_ICBAPhysicalDevice | Identifies the ICBAPhysicalDevice interface uniquely. |
| UUID_ICBABrowse | Identifies the ICBABrowse interface uniquely. |
| UUID_ICBAPersist | Identifies the ICBAPersist interface uniquely. |
| UUID_ICBALogicalDevice | Identifies the ICBALogicalDevice interface uniquely. |
| UUID_ICBAState | Identifies the ICBAState interface uniquely. |
| UUID_ICBATime | Identifies the ICBATime interface uniquely. |
| UUID_ICBAGroupError | Identifies the ICBAGroupError interface uniquely. |
| UUID_ICBAAccoMgt | Identifies the ICBAAccoMgt interface uniquely. |
| UUID_ICBAAccoServer | Identifies the ICBAAccoServer interface uniquely. |
| UUID_ICBAAccoCallback | Identifies the ICBAAccoCallback interface uniquely. |
| UUID_ICBAAccoSync | Identifies the ICBAAccoSync interface uniquely. |
| UUID_ICBARTAuto | Identifies the ICBARTAuto interface uniquely. |
| UUID_PhysicalDevice | Identifies the Physical Device class uniquely. |
| UUID_LogicalDevice | Identifies the Logical Device class uniquely. |
| UUID_ACCO | Identifies the ACCO class uniquely. |
| UUID_RTAuto | Identifies the RTAuto class uniquely. |

#### 5.3.1.9 Pointer types

##### 5.3.1.9.1 Interface pointer

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | Interface Pointer |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This data type defines a 4 octet fixed length data type.

##### 5.3.1.9.2 LPWSTR

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | LPWSTR |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This data type defines a reference to an UnicodeString.

#### 5.3.1.10 Time types

##### 5.3.1.10.1 BinaryTime0

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 40 |
| 2 | Data type Name | = | BinaryTime0 |
| 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 type has a length of two octets. The unit of time for this type is 10 µs.

##### 5.3.1.10.2 BinaryTime1

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 41 |
| 2 | Data type Name | = | BinaryTime1 |
| 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 type has a length of two octets. The unit of time for this type is 100 µs.

##### 5.3.1.10.3 BinaryTime2

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 42 |
| 2 | Data type Name | = | BinaryTime2 |
| 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 type has a length of two octets. The unit of time for this type is 1 ms.

#### 5.3.1.10.4 BinaryTime3

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 43 |
| 2 | Data type Name | = | BinaryTime3 |
| 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 type has a length of two octets. The unit of time for this type is 10 ms.

#### 5.3.1.10.5 BinaryTime4

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 44 |
| 2 | Data type Name | = | BinaryTime4 |
| 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 type has a length of four octets. The unit of time for this type is 10 µs.

#### 5.3.1.10.6 BinaryTime5

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 45 |
| 2 | Data type Name | = | BinaryTime5 |
| 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 type has a length of four octets. The unit of time for this type is 100 µs.

#### 5.3.1.10.7 BinaryTime6

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 46 |
| 2 | Data type Name | = | BinaryTime6 |
| 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 type has a length of four octets. The unit of time for this type is 1 ms.

#### 5.3.1.10.8 BinaryTime7

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 47 |
| 2 | Data type Name | = | BinaryTime7 |
| 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 type has a length of six octets. The unit of time for this type is 1 ms.

#### 5.3.1.10.9 BinaryTime8

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 48 |
| 2 | Data type Name | = | BinaryTime8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 6 |

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 six octets. The unit of time for this type is 10 µs.

#### 5.3.1.10.10 BinaryTime9

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 49 |
| 2 | Data type Name | = | BinaryTime9 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 6 |

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 binary time type has a length of six octets. The unit of time for this type is 100 µs.

#### 5.3.1.10.11 TIME

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | TIME |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This IEC 61131-3 type is a two's complement binary number with a length of four octets. The unit of time for this type is 1 ms.

#### 5.3.1.10.12 ITIME

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | ITIME |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

This IEC 61131-3 type extension is a two's complement binary number with a length of two octets. The unit of time for this type is 1 ms.

#### 5.3.1.10.13 FTIME

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | FTIME |

| | | | |
|---|---|---|---|
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

#### 5.3.1.10.14 LTIME

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | LTIME |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

This IEC 61131-3 type extension is a two's complement binary number with a length of eight octets. The unit of time for this type is 1 µs.

#### 5.3.1.10.15 NetworkTime

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 58 |
| 2 | Data type Name | = | NetworkTime |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

This data type is composed of an integer value and of an unsigned value that express the network time.

The first element is an Unsigned32 data type that provides the network time in seconds since 1900.01.01 00:00:00(UTC) for network time greater/equal 1984.01.01 00:00:00 (UTC) and less than or 2036.07.02 06:28:16(UTC), or in seconds since 2036.07.02 06:28:16(UTC) for Network time greater or equal 2036.07.02 06:28:16(UTC).

The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$s).

#### 5.3.1.10.16 NetworkTimeDifference

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 59 |
| 2 | Data type Name | = | NetworkTimeDifference |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

This data type is composed of an integer value and of an unsigned value that express the difference in network time. The first element is an Integer32 data type that provides the network time difference in seconds. The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$s.

### 5.3.1.11 VisibleString character types

#### 5.3.1.11.1 UNICODE char

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 36 |
| 2 | Data type Name | = | UnicodeChar |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

This type is defined as a single character in the UNICODE string type.

#### 5.3.1.11.2 VisibleString1

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 25 |
| 2 | Data type Name | = | VisibleString1 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 1 |

This type is defined as a single character in the ISO VisibleString type.

#### 5.3.1.11.3 VisibleString2

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 26 |
| 2 | Data type Name | = | VisibleString2 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

This type contains two elements of type VisibleString.

#### 5.3.1.11.4 VisibleString4

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 27 |
| 2 | Data type Name | = | VisibleString4 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

This type contains four elements of type VisibleString.

#### 5.3.1.11.5 VisibleString8

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 28 |
| 2 | Data type Name | = | VisibleString8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

This type contains eight elements of type VisibleString.

#### 5.3.1.11.6 VisibleString16

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 29 |
| 2 | Data type Name | = | VisibleString16 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 16 |

This type contains 16 elements of type VisibleString.

### 5.3.2 String types

#### 5.3.2.1 BitString

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 14 |
| 2 | Data type Name | = | Bitstring |
| 3 | Format | = | STRING |
| 5.1 | Octet Length | = | 1 to n |

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

#### 5.3.2.2 CompactBooleanArray

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 37 |
| 2 | Data type Name | = | CompactBooleanArray |
| 3 | Format | = | STRING |
| 6.1 | Octet Length | = | 1 |

In this type, each bit value of 0 (zero) represents the Boolean value FALSE and each bit value of 1 represents the Boolean value TRUE.

#### 5.3.2.3 CompactBCDArray

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 38 |
| 2 | Data type Name | = | CompactBCDArray |
| 3 | Format | = | STRING |
| 4.1 | Octet Length | = | 1 |

This type is used to pack an ordered series of BCD values, two per octet, into an ordered series of octets. The first octet contains the most significant BCD value in its most significant quartet. If the number of BCD values is odd, the least significant quartet of the final octet is set to the reserved value "1111".

#### 5.3.2.4 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 61158-6-5 defines the order of transmission.

#### 5.3.2.5 UNICODEString

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 39 |
| 2 | Data type Name | = | UnicodeString |
| 3 | Format | = | STRING |
| 4.1 | Octet Length | = | 2 |

This type is defined as the UNICODE string type.

#### 5.3.2.6 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.

### 5.3.3 Structure types

#### 5.3.3.1 ADDCONNECTIONIN

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | ADDCONNECTIONIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 5 |
| 5.2.1 | Field Name | = | ProviderItem |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | ConsumerItem |
| 5.2.4 | Field Data type | = | LPWSTR |
| 5.2.5 | Field Name | = | Persistence |
| 5.2.6 | Field Data type | = | PERSISTDEF |
| 5.2.7 | Field Name | = | SubstituteValue |
| 5.2.8 | Field Data type | = | VARIANT |
| 5.2.9 | Field Name | = | Epsilon |
| 5.2.10 | Field Data type | = | VARIANT |

This data type defines the ADDCONNECTIONIN data type.

This data type is structured as follows.

**ProviderItem**
This field contains the name of the source data item.

**ConsumerItem**
This field contains the name of the sink data item.

**Persistence**
This field describes the required persistence of the connection information. The values CBAVolatile and CBAPersistent are allowed.

**SubstituteValue**
This field specifies the substitute value according the data type of the connection item.

**Epsilon**
This field specifies the hysteresis as absolute change of the value.

#### 5.3.3.2 ADDCONNECTIONOUT

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | ADDCONNECTIONOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 3 |
| 5.2.1 | Field Name | = | ConsumerID |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | Version |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | ErrorState |
| 5.2.6 | Field Data type | = | HRESULT |

This data type defines the ADDCONNECTIONOUT data type.

This data type is structured as follows.

**ConsumerID**
This field contains the identifier of the consumer.

**Version**
This field contains the version number of the connection.

**ErrorState**
This field contains the error condition of the connection.

#### 5.3.3.3 BSTR

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | BSTR |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | OctetCount |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | UnicodeString |
| 5.2.4 | Field Data type | = | UnicodeString |

This data type defines an UnicodeString with preceding octet count value. The count contains the number of octets, not UNICODE characters, in the string. This count does not include the terminating zero character (2 octets).

#### 5.3.3.4 CONNECTIN

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | CONNECTIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | ProviderItem |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | DataType |
| 5.2.4 | Field Data type | = | VARTYPE |
| 5.2.5 | Field Name | = | Epsilon |
| 5.2.6 | Field Data type | = | VARIANT |
| 5.2.7 | Field Name | = | ConsumerID |
| 5.2.8 | Field Data type | = | unsigned long |

This data type defines the CONNECTIN data type.

This data type is structured as follows.

**ProviderItem**
This field contains the name of the source data item.

**DataType**
This field contains the type code for the data item.

**Epsilon**
This field specifies the hysteresis as absolute change of the value.

**ConsumerID**
This field contains the identifier of the consumer.

#### 5.3.3.5 CONNECTOUT

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | CONNECTOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |

| | | | |
|---|---|---|---|
| 5.2.1 | Field Name | = | ProviderID |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | ErrorState |
| 5.2.4 | Field Data type | = | HRESULT |

This data type defines the CONNECTOUT data type.

This data type is structured as follows.

**ProviderID**
This field contains the identifier of the provider.

**ErrorState**
This field contains the error condition of the interconnection.

#### 5.3.3.6 EXCEPTINFO

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | EXCEPINFO |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 9 |
| 5.2.1 | Field Name | = | wCode |
| 5.2.2 | Field Data type | = | unsigned short |
| 5.2.3 | Field Name | = | wReserved |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | bstrSource |
| 5.2.6 | Field Data type | = | BSTR |
| 5.2.7 | Field Name | = | bstrDescription |
| 5.2.8 | Field Data type | = | BSTR |
| 5.2.9 | Field Name | = | bstrHelpFile |
| 5.2.10 | Field Data type | = | BSTR |
| 5.2.11 | Field Name | = | dwHelpContext |
| 5.2.12 | Field Data type | = | unsigned long |
| 5.2.13 | Field Name | = | pvReserved |
| 5.2.14 | Field Data type | = | unsigned long |
| 5.2.15 | Field Name | = | pfnDeferredFillIn |
| 5.2.16 | Field Data type | = | unsigned long |
| 5.2.17 | Field Name | = | scode |
| 5.2.18 | Field Data type | = | long |

This data type defines the EXCEPINFO data type to describe an exception that occurred during the Invoke service.

This data type is structured as follows.

**wCode**
This field contains an error code identifying the error. Error codes should be greater than 1000. Either this field or the scode field should be filled in; the other should be set to 0.

**wReserved**
This field is reserved and should be set to 0.

**bstrSource**
This field contains a textual, human-readable name of the source of the exception. Typically, this is an application name. This field should be filled in by the implementor of the IDispatch interface.

**bstrDescription**
This field contains a textual, human-readable description of the error intended for the customer. If no description is available, use Null.

**bstrHelpFile**
This field contains the fully qualified drive, path, and file name of a Help file with more information about the error. If no Help is available, use Null.

**dwHelpContext**
This field contains the Help context ID of the topic within the Help file. This field should be filled in if and only if the bstrHelpFile field is not Null.

**pvReserved**
This field is reserved and should be set to Null.

**pfnDeferredFillIn**
This field contains the address of a function that takes an EXCEPINFO structure as an argument and returns an HRESULT value. If deferred, fill-in is not desired, this field should be set to Null.

**scode**
This field contains a return value describing the error. Either this field or the wCode field should be filled in; the other should be set to 0.

#### 5.3.3.7 DATE_AND_TIME

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | DATE_AND_TIME |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Time_Of_Day_Element |
| 5.2.2 | Field Data type | = | TIME_OF_DAY |
| 5.3.1 | Field Name | = | Date_Element |
| 5.3.2 | Field Data type | = | DATE |

This IEC 61131-3 type extension is a structure which expresses both the date (as a number of days starting from January 1st, 1972 until June 6th, 2151),and the time of day as a number of ms starting from midnight.

#### 5.3.3.8 FILETIME

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | FILETIME |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | LowDataTime |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | HighDataTime |
| 5.2.4 | Field Data type | = | unsigned long |

This data type is a 64 bit time value representing the number of 100-nanosecond intervals since January 1, 1601.

#### 5.3.3.9 GETIDOUT

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | GETIDOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2.1 | Field Name | = | ConsumerID |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | State |
| 5.2.4 | Field Data type | = | VT_BOOLEAN |
| 5.2.5 | Field Name | = | Version |
| 5.2.6 | Field Data type | = | unsigned short |
| 5.2.7 | Field Name | = | ErrorState |
| 5.2.8 | Field Data type | = | HRESULT |

This data type defines the GETIDOUT data type.

This data type is structured as follows.

**ConsumerID**
This field contains the identifier of the consumer.

**State**
This field contains the state of the connection. The value TRUE indicates an active and the value FALSE an inactive connection.

**Version**
This field contains the version number of the connection.

**ErrorState**
This field contains the error condition of the connection.

#### 5.3.3.10 GETCONNECTIONOUT

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | GETCONNECTIONOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 10 |
| 5.2.1 | Field Name | = | Provider |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | ProviderItem |
| 5.2.4 | Field Data type | = | LPWSTR |
| 5.2.5 | Field Name | = | ConsumerItem |
| 5.2.6 | Field Data type | = | LPWSTR |
| 5.2.7 | Field Name | = | SubstituteValue |
| 5.2.8 | Field Data type | = | VARIANT |
| 5.2.9 | Field Name | = | Epsilon |
| 5.2.10 | Field Data type | = | VARIANT |
| 5.2.11 | Field Name | = | QoSType |
| 5.2.12 | Field Data type | = | unsigned short |
| 5.2.13 | Field Name | = | QoSValue |
| 5.2.14 | Field Data type | = | unsigned short |
| 5.2.15 | Field Name | = | State |
| 5.2.16 | Field Data type | = | VT_BOOLEAN |
| 5.2.17 | Field Name | = | Persistence |
| 5.2.18 | Field Data type | = | PERSISTDEF |
| 5.2.19 | Field Name | = | Version |
| 5.2.20 | Field Data type | = | unsigned short |
| 5.2.21 | Field Name | = | ErrorState |
| 5.2.22 | Field Data type | = | HRESULT |

This data type defines the GETCONNECTIONOUT data type.

This data type is structured as follows.

**Provider**
This field contains the name of the providers LDev (Format: PDev!LDev).

**ProviderItem**
This field contains the name of the source data item.

**ConsumerItem**
This field contains the name of the sink data item.

**SubstituteValue**
This field specifies the substitute value according the data type of the connection item.

**Epsilon**
This field specifies the hysteresis as absolute change of the value.

**QoSType**
This field contains the quality of service type.

**QoSValue**
This field contains the quality of service qualifier.

**State**
This field contains the state of the connection. The value TRUE indicates an active and the value FALSE an inactive connection.

**Persistence**
This field describes the required persistence of the connection information. The values CBAVolatile and CBAPersistent are allowed.

**Version**
This field contains the version number of the connection.

**ErrorState**
This field contains the error condition of the connection.

#### 5.3.3.11 QualifiedOctetString2

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | QualifiedOctetString2 |
| 5 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Value |
| 5.2.2 | Field Data type | = | Octet String(2) |
| 5.2.3 | Field Name | = | Status |
| 5.2.4 | Field Data type | = | Unsigned8 |

This data type defines the QualifiedOctetString2.

This data type is structured as follows.

**Value**
This field contains the value as Octet String with the length of 2.

**Status**
This field describes the status of the value as a qualifier.

#### 5.3.3.12 QualifiedFloat32

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | QualifiedFloat32 |
| 5 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Value |
| 5.2.2 | Field Data type | = | Float32 |
| 5.2.3 | Field Name | = | Status |
| 5.2.4 | Field Data type | = | Unsigned8 |

This data type defines the QualifiedFloat32.

This data type is structured as follows.

**Value**
This field contains the value as Float32.

**Status**
This field describes the status of the value as a qualifier.

#### 5.3.3.13 QualifiedUnsigned8

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | QualifiedUnsigned8 |
| 5 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Value |
| 5.2.2 | Field Data type | = | Unsigned8 |
| 5.2.3 | Field Name | = | Status |
| 5.2.4 | Field Data type | = | Unsigned8 |

This data type defines the QualifiedUnsigned8.

This data type is structured as follows.

**Value**
This field contains the value as Unsigned8.

**Status**
This field describes the status of the value as a qualifier.

#### 5.3.3.14 READITEMOUT

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | READITEMOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2.1 | Field Name | = | Value |
| 5.3.2 | Field Data type | = | VARIANT |
| 5.4.3 | Field Name | = | QualityCode |
| 5.5.4 | Field Data type | = | ITEMQUALITYDEF |
| 5.6.5 | Field Name | = | TimeStamp |
| 5.7.6 | Field Data type | = | FILETIME |
| 5.8.7 | Field Name | = | ErrorState |
| 5.9.8 | Field Data type | = | HRESULT |

This data type defines the READITEMOUT data type.

This data type is structured as follows.

**Value**
This field contains the value itself with the format according to the appropriate data type.

**QualityCode**
This field contains the Quality Code of the data value.

**TimeStamp**
This field contains the time stamp of the value.

**ErrorState**
This field contains the error condition of the connection.

#### 5.3.3.15 SAFEARRAY

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | RGSABOUND |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Elements |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | Left Bound |
| 5.2.4 | Field Data type | = | long |

This data type is a helper describing one array element with boundary information needed to define the SAFEARRAY below.

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | SAFEARRAY |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 6 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Dims |
| 5.2.2 | Field Data type | = | unsigned short |
| 5.2.3 | Field Name | = | Features |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | Elements |
| 5.2.6 | Field Data type | = | unsigned long |
| 5.2.7 | Field Name | = | Locks |
| 5.2.8 | Field Data type | = | unsigned long |
| 5.2.9 | Field Name | = | Data Address |
| 5.2.10 | Field Data type | = | unsigned long |
| 5.2.11 | Field Name | = | Rgsabound |
| 5.2.12 | Format | = | ARRAY |
| 5.2.12.1 | Number of Array Elements | = | Dims |
| 5.2.12.2 | Array Element Data type | = | RGSABOUND |

This data type expresses a one- or multi-dimensional array of a single data type. (However, this single data type can be a VARIANT, allowing you arrays of mixed types.) The reason these arrays are called safe arrays is because they contain bounds information, allowing to check the index or indices against the bounds before accessing the data in the array. The lower bound of the array does not have to be zero, so the safe array has to store its lower bound as well as the size. Finally, safe arrays allow locking (and unlocking) so it can be sure the pointer to the data you get is valid.

This data type is structured as follows.

**Dims**
This field contains the dimension of the SAFEARRAY.

**Features**
This field contains flags for allocation type and data type of the SAFEARRAY.

**Elements**
This field contains the size of a single element of the SAFEARRAY.

**Locks**
This field contains the lock counter of the SAFEARRAY.

**Data Address**
This field contains the address of the actual data of the SAFEARRAY.

**Rgsabound**
This field contains an array with the information about boundaries for each dimension of the SAFEARRAY. Therefore, the number of array elements is according to the dimension of the safe array

#### 5.3.3.16 SHORT_STRING

| **CLASS:** | | **Data type** | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | SHORT_STRING |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Charcount_Element |
| 5.2.2 | Field Data type | = | USINT |

| | | | |
|---|---|---|---|
| 5.3.1 | Field Name | = | Stringcontents_Element |
| 5.3.2 | Field Data type | = | OctetString |

This IEC 61131-3 type extension is composed of two elements. Charcount_Element gives the current number of characters in the Stringcontents_Element (one USINT per character).

#### 5.3.3.17 STRING

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | STRING |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Charcount_Element |
| 5.2.2 | Field Data type | = | UINT |
| 5.3.1 | Field Name | = | Stringcontents_Element |
| 5.3.2 | Field Data type | = | OctetString |

This IEC 61131-3 type is composed of two elements. Charcount_Element gives the current number of characters in the Stringcontents_Element (one USINT per character).

#### 5.3.3.18 STRING2

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | STRING2 |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Charcount_Element |
| 5.2.2 | Field Data type | = | UINT |
| 5.3.1 | Field Name | = | String2contents_Element |
| 5.3.2 | Field Data type | = | OctetString |

This IEC 61131-3 data type extension is composed of two elements. Charcount_Element gives the current number of characters in the String2contents_Element (one UINT per character).

#### 5.3.3.19 STRINGN

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | not used |
| 2 | Data type Name | = | STRINGN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 3 |
| 5.2.1 | Field Name | = | Charsize_Element |
| 5.2.2 | Field Data type | = | UINT |
| 5.3.1 | Field Name | = | Charcount_Element |
| 5.3.2 | Field Data type | = | UINT |
| 5.4.1 | Field Name | = | StringNcontents_Element |
| 5.4.2 | Field Data type | = | OctetString |

This IEC 61131-3 type extension is composed of three elements. Charsize_Element gives the size of a character in StringNcontents_Element (N = number of USINT). Charcount_Element gives the current number of characters in the StringNcontents_Element (N USINT per character).

#### 5.3.3.20 VARIANT

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | VARIANT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 5 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Tag |
| 5.2.2 | Field Data type | = | VARTYPE |
| 5.2.3 | Field Name | = | Padding1 |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | Padding2 |
| 5.2.6 | Field Data type | = | unsigned short |
| 5.2.7 | Field Name | = | Padding3 |
| 5.2.8 | Field Data type | = | unsigned short |
| 5.2.9 | Field Name | = | Value |
| 5.2.10 | Field Data type | = | see Table 9 |

This data type expresses a VARIANT that containing possible values of data types according to Table 9. The Tag field contains the numeric identifier of the data type to identify the data type of the value. The overall size of the VARIANT is 16 octets.

**Table 9 – Data type names for value**

| Data type |
|---|
| VT_BOOLEAN |
| char |
| short |
| long |
| unsigned char |
| unsigned short |
| unsigned long |
| float |
| double |
| date |
| currency |
| Address of a BSTR |
| Address of a SAFEARRAY |
| Address of a userdefined struct |
| Interface Pointer |
| HRESULT |

#### 5.3.3.21 WRITEITEMIN

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | WRITEITEMIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Item |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | Value |
| 5.2.4 | Field Data type | = | VARIANT |

This data type defines the WRITEITEMIN data type.

This data type is structured as follows.

**Item**
This field contains the name of the data item.

**Value**
This field contains the value itself with the format according to the appropriate data type.

#### 5.3.3.22 WRITEITEMQCDIN

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | WRITEITEMQCDIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 3 |
| 5.2.1 | Field Name | = | WriteItem |
| 5.2.2 | Field Data type | = | WRITEITEMIN |
| 5.2.3 | Field Name | = | QualityCode |
| 5.2.4 | Field Data type | = | ITEMQUALITYDEF |
| 5.2.5 | Field Name | = | TimeStamp |
| 5.2.6 | Field Data type | = | FILETIME |

This data type defines the WRITEITEMQCDIN data type.

This data type is structured as follows.

**WriteItem**
This field contains the name and value of the data item.

**QualityCode**
This field contains the Quality Code of the data value.

**TimeStamp**
This field contains the time stamp of the value.

#### 5.3.3.23 DISPPARAMS

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | UUID |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2.1 | Field Name | = | Data1 |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | Data2 |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | Data3 |
| 5.2.6 | Field Data type | = | unsigned short |
| 5.2.7 | Field Name | = | Data4 |
| 5.2.8.1 | Format | = | ARRAY |
| 5.2.8.4.1 | Number of Array Elements | = | 8 |
| 5.2.8.4.2 | Array Element Data type | = | unsigned char |

This data type defines a 16 octet fixed length data type. The semantic is specified by the used ORPC model and beyond the scope of this standard.

This data type is structured as follows.

**Data1**
This field contains the first eight hexadecimal digits of the UUID.

**Data2**
This field contains the first group of four hexadecimal digits of the UUID.

**Data3**
This field contains the second group of four hexadecimal digits of the UUID.

**Data4**
This field contains an array of eight elements. The first two elements contain the third group of four hexadecimal digits of the UUID. the remaining six elements contain the final 12 hexadecimal digits of the UUID.

Predefined values for TYPE 10 items are shown in Table 10.

**Table 10 – UUID**

| Value | Description |
|---|---|
| UUID_NULL | |
| UUID_IUnknown | Identifies the IUnknown interface uniquely. |
| UUID_IDispatch | Identifies the IDispatch interface uniquely. |
| UUID_ICBAPhysicalDevice | Identifies the ICBAPhysicalDevice interface uniquely. |
| UUID_ICBABrowse | Identifies the ICBABrowse interface uniquely. |
| UUID_ICBAPersist | Identifies the ICBAPersist interface uniquely. |
| UUID_ICBALogicalDevice | Identifies the ICBALogicalDevice interface uniquely. |
| UUID_ICBAState | Identifies the ICBAState interface uniquely. |
| UUID_ICBATime | Identifies the ICBATime interface uniquely. |
| UUID_ICBAGroupError | Identifies the ICBAGroupError interface uniquely. |
| UUID_ICBAAccoMgt | Identifies the ICBAAccoMgt interface uniquely. |
| UUID_ICBAAccoServer | Identifies the ICBAAccoServer interface uniquely. |
| UUID_ICBAAccoCallback | Identifies the ICBAAccoCallback interface uniquely. |
| UUID_ICBAAccoSync | Identifies the ICBAAccoSync interface uniquely. |
| UUID_ICBARTAuto | Identifies the ICBARTAuto interface uniquely. |
| UUID_PhysicalDevice | Identifies the Physical Device class uniquely. |
| UUID_LogicalDevice | Identifies the Logical Device class uniquely. |
| UUID_ACCO | Identifies the ACCO class uniquely. |
| UUID_RTAuto | Identifies the RTAuto class uniquely. |

### 5.4 Data type ASE service specification

There are no operational services defined for the type object.

## 6 Communication model specification

### 6.1 Concepts

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

### 6.2 ASEs

#### 6.2.1 Object management ASE

##### 6.2.1.1 Overview

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

To dynamically create or delete the network view of an APO, or to access its attribute values, this FAL ASE defines common FAL APO management services. Each APO Class definition specifies which of these services may be included in the definitions of APOs of the class. These services operate using the client/server model.

#### 6.2.1.2 FAL management model specification

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

#### 6.2.1.3 FAL management model services

##### 6.2.1.3.1 Supported services

Management services are defined for managing APs and APOs. For simplicity APs and APOs are referred to as objects in the remainder of 6.2.1.3.

Create and Delete services are specified for a class if the network view of an object of the class can be dynamically created or deleted. Get Attributes and Set Attributes services are specified to indicate when its attributes can be read or written. The attributes of an APO which can be set 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 Begin Set Attributes and End Set Attributes services are defined. The Begin Set Attributes service is used to request the remote AP for permission to change its object definitions. The End Set Attributes service is used to indicate to the remote AP that changes to its object definitions are complete and that it can resume normal operation.

##### 6.2.1.3.2 Create service

###### 6.2.1.3.2.1 Service overview

This confirmed service is used to make an object visible and accessible across the network. As part of the service, initial values for the object's attributes may be specified. If all attribute values to be initialized cannot be conveyed in a single APDU, then the Set Attributes service can be used once the object has been created for attributes that may be updated using the Set Attributes service. The scope of visibility for created objects is Application Process specific.

The List of Supported Attributes parameter is used to indicate to the requester the attributes which were actually created for the object. If initial values were supplied for attributes that were not created, the List of Supported Attributes can be used to determine the ones which were not created and initialized according to the request. This capability represents a form of negotiation when creating an object.

NOTE Whether or not an AP actually creates the object is a local matter. The intent of the service is to request that it be prepared for remote access.

###### 6.2.1.3.2.2 Service parameters

The service parameters for this service are shown in Table 11.

**Table 11 – Create service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| FAL ASE / FAL Class | M | M (=) | | |
| List of Attribute IDs and Initial Values | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | M (=) |
| Numeric ID | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

**Argument**

The argument contains the parameters of the service request.

**List of Attribute IDs and Initial Values**

This parameter identifies and specifies the initial value of one or more of the object's attributes to be set during the creation process. At least one of the object's key attributes should be present. Values for mandatory and optional attributes may be supplied. The AP that creates the object determines the complement of attributes supported for the object.

NOTE Initial values are only required for mandatory attributes if the AP creating the object does not have the ability to supply an initial value on its own. How it acquires these values on its own is beyond the scope of this standard.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.1.3.2.3 Service procedure

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

### 6.2.1.3.3 Delete service

#### 6.2.1.3.3.1 Service overview

This confirmed service is used to make an object that is currently visible and accessible not visible and not accessible across the network.

NOTE Whether or not the real object is actually deleted or simply removed from network accessibility is a local matter.

#### 6.2.1.3.3.2 Service parameters

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

**Table 12 – Delete service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| FAL ASE / FAL Class | M | M (=) | | |
| Key Attribute | M | M (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**FAL ASE/FAL Class**

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

**Key Attribute**

This parameter identifies the object by one of its key attributes.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.1.3.3.3 Service procedure

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

Once deleted, an object is no longer accessible using the services of the FAL.

### 6.2.1.3.4 Get attributes service

#### 6.2.1.3.4.1 Service overview

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

#### 6.2.1.3.4.2 Service parameters

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

**Table 13 – Get attributes service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List of Objects and their Attributes | S | S (=) | | |
| Key Attribute | M | M (=) | | |
| List of Requested Attributes | M | M (=) | | |
| Data type Requested | C | C (=) | | |
| Range of Objects and their Attributes | S | S (=) | | |
| Starting Numeric ID | M | M (=) | | |
| List of Requested Attributes | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| More | | | M | M (=) |
| List of Responses | | | M | M (=) |
| List of Supported Attributes | | | C | C (=) |
| Error Status | | | S | S (=) |
| List of Attribute Values | | | S | S (=) |
| Data type Description | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

### Argument

The argument contains the parameters of the service request.

### List of Objects and their Attributes

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

#### Key Attribute

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

#### List of Requested Attributes

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

### Range of Objects and their Attributes

This selection type parameter provides a range of numeric identifiers for objects of a single class, and optionally for a series of objects of different classes, and the attributes that are being requested. Three values for identifying attributes are possible, all attributes, mandatory attributes only, or a list of individual attributes. It is present if the user is requesting attributes for objects within a given range of numeric ids.

**Starting Numeric ID**

This parameter specifies the first in a range of numeric ids. The attributes being requested are for all objects beginning with and after this numeric id. The value ALL may be used to indicate that attributes for all objects for the specified class are to be returned.

NOTE Multiple requests may have to be issued with successively higher numeric ids when not all the desired object definitions can be returned in a single response.

**List of Requested Attributes**

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

**Data type Requested**

This conditional parameter is present only when the attributes of a variable object are being requested. It indicates whether the data type description is to be returned with requested variable attributes.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**More**

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

**List of Responses**

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

**List of Supported Attributes**

This parameter identifies the attributes supported by the responder for the specified object. It is present if the specified object exists.

**Error Status**

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

**List of Attribute Values**

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

**Data type Description**

This conditional parameter is present only when data type description for a variable object has been requested. It contains the complete data type description for the variable.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.1.3.4.3 Service procedure

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

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

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

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

### 6.2.1.3.5 Set attributes service

#### 6.2.1.3.5.1 Service overview

This confirmed service is used to update the current value of a list of attributes for one or more objects. This service operates in an all or nothing fashion — if all the requested attributes for all the objects cannot be set, the service fails.

NOTE This service may be used to load entire object definitions by indicating that all mandatory attributes are to be updated.

#### 6.2.1.3.5.2 Service parameters

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

**Table 14 – Set attributes service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List of Object Definitions | M | M (=) | | |
| FAL ASE / FAL Class | M | M (=) | | |
| Key Attribute | M | M (=) | | |
| List of Attribute Updates | M | M (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| List of Supported Attributes | | | C | C (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**List of Object Updates**

This parameter specifies the list of objects and their attributes that are to be updated.

**FAL ASE/FAL Class**

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

**Key Attribute**

This parameter identifies the object by one of its key attributes.

**List of Attribute Updates**

This parameter identifies one or more of the attributes that are to be updated and their values.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

**List of Supported Attributes**

This conditional parameter identifies the attributes that are supported by the object. It is present when a request was received to update unsupported attributes.

#### 6.2.1.3.5.3 Service procedure

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

The Set Attributes Service is an "all or nothing" service. All or nothing means that the service succeeds only if all the requested updates succeed.

### 6.2.1.3.6 Begin set attributes service

#### 6.2.1.3.6.1 Service overview

The Begin Set Attributes prepares the AP for updates to its object definitions, such that the updates are free of interaction or not free of interaction. The AP returns a successful response to indicate to the requester that it may begin updating object definitions.

#### 6.2.1.3.6.2 Service parameters

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

**Table 15 – Begin set attributes**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Consequence | M | M (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |

**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 updates to individual object definitions, or not free of interaction with completely new load of all object definitions.

- loading free of interaction
- reload, not free of interaction
- new load, not free of interaction

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.1.3.6.3 Service procedure

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

### 6.2.1.3.7 End set attributes service

#### 6.2.1.3.7.1 Service overview

The End Set Attributes service terminates the process of updating object definitions. It is used to indicate to the AP that all object definition updates have been completed.

#### 6.2.1.3.7.2 Service parameters

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

**Table 16 – End set attributes**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Numeric ID | | | C | C (=) |

**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.

**Numeric ID**

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

#### 6.2.1.3.7.3 Service procedure

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

### 6.2.2 Application process ASE

#### 6.2.2.1 Overview

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

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

#### 6.2.2.2 AP class specification

##### 6.2.2.2.1 Formal model

The AP class specifies the attributes and services defined for application processes. Its parent class "top" indicates the top of the FAL class tree. The numeric identifier for the AP Object is always 0. The use of this reserved value permits management services to be used for AP objects without knowing their names.

**FAL ASE:** **AP ASE**
**CLASS:** AP
**CLASS ID:** 16
**PARENT CLASS:** TOP

**IDENTIFY, GET STATUS, and STATUS NOTIFICATION SERVICE ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | (m) | Attribute: | List Of AP Service Attributes |
| 1.1 | (m) | Attribute: | Manufacturer Identifier |
| 1.1.1 | (m) | Attribute: | Vendor Name |
| 1.1.2 | (m) | Attribute: | Model Identifier |
| 1.1.3 | (m) | Attribute: | Vendor Revision |
| 1.1.4 | (o) | Attribute: | Serial Number |
| 1.2 | (o) | Attribute: | AP Descriptor Reference |
| 1.3 | (m) | Attribute: | Network Access State |
| 1.3.1 | (m) | Attribute: | Service Access Status |
| 1.3.2 | (m) | Attribute: | Operational Status |

**SYSTEM MANAGEMENT ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | (m) | Attribute: | List Of System Management Attributes |
| 2.1 | (m) | Attribute: | List Of AR Endpoint Info |
| 2.1.1 | (m) | Attribute: | Numeric Id |
| 2.1.2 | (m) | Attribute: | Configured Max FAL PDU Size Sending |
| 2.1.3 | (m) | Attribute: | Configured Max FAL PDU Size Receiving |
| 2.1.4 | (m) | Attribute: | Configured Max Outstanding Services Requesting |
| 2.1.5 | (m) | Attribute: | Configured Max Outstanding Services Responding |
| 2.1.6 | (m) | Attribute: | List of Supported Services |
| 2.1.7 | (o) | Attribute: | Configured Max Data Structure Nesting Level |
| 2.2 | (m) | Attribute: | List Of In Use AR Endpoint Info |
| 2.2.1 | (m) | Attribute: | Context State |
| 2.2.2 | (m) | Attribute: | Negotiated Max FAL PDU Size Sending |
| 2.2.3 | (m) | Attribute: | Negotiated Max FAL PDU Size Receiving |
| 2.2.4 | (m) | Attribute: | Negotiated Max Outstanding Services Requesting |
| 2.2.5 | (m) | Attribute: | Negotiated Max Outstanding Services Responding |
| 2.2.6 | (m) | Attribute: | Outstanding Services Requesting Counter |
| 2.2.7 | (m) | Attribute: | Outstanding Services Responding Counter |
| 2.2.8 | (o) | Attribute: | Negotiated Max Data Structure Nesting Level |
| 2.2.9 | (o) | Attribute: | Negotiated List of Supported Services |

**OBJECT MANAGEMENT ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 3 | (m) | Attribute: | List Of Managed AP Attributes |
| 3.1 | (m) | Attribute: | List Of Supported APO Classes and Objects |
| 3.1.1 | (m) | Attribute: | List Header |
| 3.1.1.1 | (o) | Attribute: | Numeric Identifier = 0 |
| 3.1.1.2 | (m) | Attribute: | ROM / RAM Flag (TRUE, FALSE) |
| 3.1.1.3 | (m) | Attribute: | Max Name Length |
| 3.1.1.4 | (m) | Attribute: | Access Protection Supported (TRUE, FALSE) |
| 3.1.1.5 | (m) | Attribute: | Version Of List |
| 3.1.1.6 | (m) | Attribute: | List Local Reference |
| 3.1.2 | (c) | Constraint | Data type Supported |
| 3.1.2.1 | (o) | Attribute: | Numeric Identifier = 1 |
| 3.1.2.2 | (m) | Attribute: | Number Of Data type Objects |
| 3.1.2.3 | (m) | Attribute: | Local Reference |
| 3.1.3 | (c) | Constraint | Variable \|\| Load Region \|\| Event Supported |
| 3.1.3.1 | (o) | Attribute: | Static Object Numeric Identifier |
| 3.1.3.2 | (m) | Attribute: | Number Of Static Objects (Variables + Load Region + Events) |
| 3.1.3.3 | (m) | Attribute: | Static Object Local Reference |
| 3.1.4 | (c) | Constraint | Variable List Supported |
| 3.1.4.1 | (o) | Attribute: | Numeric Identifier |

| | | | |
|---|---|---|---|
| 3.1.4.2 | (m) | Attribute: | Number Of Variable List Objects |
| 3.1.4.3 | (m) | Attribute: | Variable List Local Reference |
| 3.1.5 | (c) | Constraint | Function Invocation Supported |
| 3.1.5.1 | (o) | Attribute: | Function Invocation Numeric Identifier |
| 3.1.5.2 | (m) | Attribute: | Number Of Function Invocation Objects |
| 3.1.5.3 | (m) | Attribute: | Function Invocation Local Reference |
| 3.1.6 | (c) | Constraint | Dynamic Load Region |
| 3.1.6.1 | (o) | Attribute: | Load Region Numeric Identifier |
| 3.1.6.2 | (m) | Attribute: | Number of Load Region Objects |
| 3.1.6.3 | (m) | Attribute: | Load Region Local Reference |
| 3.2 | (o) | Attribute: | List Of DLSAP Addresses |
| 3.3 | (o) | Attribute: | List of Event Summary Selection Criteria |

**SUPPORTED ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 5 | (o) | Attribute: | List Of Supported Attributes |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | Ops Service: | Subscribe |
| 2 | (o) | Ops Service: | Identify |
| 3 | (o) | Ops Service: | Get Status |
| 4 | (o) | Ops Service: | Status Notification |
| 5 | (o) | Ops Service: | Initiate |
| 6 | (o) | Ops Service: | Terminate |
| 7 | (o) | Ops Service: | Conclude |
| 8 | (o) | Ops Service: | Reject |

#### 6.2.2.2.2 Identify, get status, and status notification service attributes

**List Of AP Service Attributes**

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

**Manufacturer Identifier**

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

**Vendor Name**

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

**Model Identifier**

This attribute specifies the model of the AP.

**Vendor Revision**

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

**Serial Number**

This optional attribute specifies the serial number of the AP.

NOTE The serial number may be used as a qualifier to uniquely identify an AP.

**AP Descriptor Reference**

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

**Network Access State**

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

**Service Access Status**

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

READY FOR COMMUNICATION All services may be used normally.

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

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

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

| | |
|---|---|
| Abort | Get Attributes |
| Status | Set Attributes |
| Identify | End Set Attributes |

**OperationalStatus**

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

OPERATIONAL

PARTIALLY OPERATIONAL

NOT OPERATIONAL

NEEDS MAINTENANCE

#### 6.2.2.2.3 System management attributes

**List Of System Management Attributes**

The following attributes are accessible through System Management. They are used by all ASEs of the AP to enforce confirmed service flow control (see the state machines of IEC 61158-6-5). They are not otherwise accessible.

**List of AR Endpoint Info**

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

**Numeric ID**

This attribute specifies the numeric id for the AREP.

**Configured Max FAL PDU Size Sending**

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

**Configured Max FAL PDU Size Receiving**

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

**Configured Max Outstanding Services Requesting (ConfigMaxOsReq)**

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

**Configured Max Outstanding Services Responding (ConfigMaxOsRsp)**

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

**List of Supported Services**

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

**Configured Max Data Structure Nesting Level (ConfigMaxDSNestingLevel)**

This optional attribute specifies the maximum number of nesting levels configured for the AR. It is present only when more than one nesting level is supported. That is, the default value for this parameter is one.

**List of In-Use AR Endpoint Info**

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

**Context State**

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

- CLOSED
- OPEN
- OPENING-REQUESTING
- OPENING-RESPONDING

**Negotiated Max FAL PDU size sending (NegMaxPduSnd)**

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

**Negotiated Max FAL PDU size receiving (NegMaxPduRcv)**

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

**Negotiated Max outstanding services requesting (NegMaxOsReq)**

This attribute specifies the maximum number of outstanding confirmed services allowed for this AREP as a client. Its value is negotiated during establishment of the AR to represent the lower value of the local Configured Max Outstanding Services Requesting attribute and the remote Configured Max Outstanding Services Responding attribute.

**Negotiated Max outstanding services responding (NegMaxOsRsp)**

This attribute specifies the maximum number of outstanding confirmed services allowed for this AREP as a server. Its value is negotiated during establishment of the AR to represent the lower value of the local Configured Max Outstanding Services Responding attribute and the remote Configured Max Outstanding Services Requesting attribute.

**Outstanding services requesting counter (OsReqCtr)**

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

**Outstanding services responding counter (OsRspCtr)**

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

**Negotiated max data structure nesting level (NegConfigMaxDSNestingLevel)**

This optional attribute specifies the maximum number of nesting levels negotiated for the AR. It is present only when more than one nesting level is supported. Its value is negotiated as the lower value configured for the two communicating APs.

**Negotiated list of supported services**

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

#### 6.2.2.2.4 Object management attributes

**List Of Managed AP Attributes**

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

**List of Supported APO Classes and Objects**

This attribute specifies the Object Definitions maintained by the AP.

**List Header**

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

**Numeric ID**

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

**ROM / RAM Flag**

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

**Max Name Length**

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

**Access Protection Supported**

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

**Version Of List**

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

**List Local Reference**

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

**XXX Object Class Supported Constraint**

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

**XXX Numeric ID**

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

**Number of XXX Entries**

This attribute specifies the number of objects in the series.

**XXX Local Reference**

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

**List of Supported APO Classes and Objects**

This attribute specifies the Object Definitions maintained by the AP.

**List of DLSAP Addresses**

This attribute specifies the available DLSAP-Addresses usable by the AP to locate its AREPs.

NOTE 1 DLSAP Address structure is defined in IEC 61158-3-1 and IEC 61158-4-1.

**List of APO Classes Supported**

This attribute specifies the classes of APOs supported by the AP.

**List of Event Summary Selection Criteria**

This optional attribute specifies a set of Boolean selection criteria that the AP is capable of using to select events when performing event summary processing. A set of selection criteria is specified by this standard. When an AP is defined, criteria specified for this attribute may be selected from this set, or they may defined specifically for the AP. This list of criteria specifies

ENABLEDWhen TRUE, event reporting for event objects is enabled

When FALSE, event reporting for event objects is disabled

DETECTED When TRUE, event occurrence has been detected

When FALSE, event occurrence has not been detected

ACKED When TRUE, event acknowledgment has been received

When FALSE, event acknowledgment is outstanding

ACTIVE When TRUE, event detection is active for the event object

When FALSE, event detection is not active for the event object

NOTE 2 Additional values may be specified by this standard in the future or they may be user defined.

#### 6.2.2.2.5 Supported attributes

**List of Supported Attributes**

This optional attribute specifies the attributes supported by the object. This list contains, at a minimum, the mandatory attributes for the class of the object.

#### 6.2.2.2.6 Services

**Subscribe**

This optional service is used by push subscribers to indicate to push publishers their requirements for receiving published data.

**Identify**

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

**Get Status**

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

**Status Notification**

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

**Initiate**

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

**Terminate**

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

**Conclude**

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

**Reject**

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

#### 6.2.2.3 Application process ASE service specification

##### 6.2.2.3.1 Supported services

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

- Subscribe
- Identify
- Get Status
- Status Notification
- Initiate
- Terminate
- Conclude
- Reject

##### 6.2.2.3.2 Subscribe service

###### 6.2.2.3.2.1 Service overview

This confirmed service is used by a push subscriber wishing to dynamically request a push publisher for the publication of selected data. The service response contains sufficient information for the subscriber to assign an AREP to receive the data.

Upon receiving a Subscribe service indication, a push publisher may initiate the publication of an object, or modify the publication constraints of a published object. The actions and policies taken by an AP to satisfy a Subscribe indication are determined by the AP itself, and are outside the scope of this standard.

This service is also used by a push subscriber to indicate to the push publisher that it is no longer subscribing to the published object.

NOTE 1 There may or may not be other subscribers who may also receive the requested published data.

NOTE 2 Publishers do not necessarily need to receive such a request to publish. For example, a publisher could be pre-configured with the knowledge of what to publish.

###### 6.2.2.3.2.2 Service primitives

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

**Table 17 – Subscribe service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Application Relationship | M | M (=) | | |
| Invoke ID | U | | | |
| Joining | S | S (=) | | |
| Published Object | M | M (=) | | |
| Schedule Interval | M | M (=) | | |
| Maximum ARs | U | U (=) | | |
| Maximum Subscriber AP Permissible Jitter | U | U (=) | | |
| Desired Publishing Start Time | U | U (=) | | |
| Earliest Publishing Start Time | U | U (=) | | |
| Latest Publishing Start Time | U | U (=) | | |
| Leaving | S | S (=) | | |
| List Of Leaving AREPs | M | M | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Data type | | | C | C (=) |
| Data Length | | | C | C (=) |
| Dedicated | | | C | C (=) |
| List Of AR Information | | | C | C (=) |
| Publishing AREP ID | | | M | M (=) |
| DL Mapping Reference | | | M | M (=) |
| Scheduled Start Time | | | M | M (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

### Argument

The argument contains the parameters of the service request.

### Joining

This parameter is selected when an AP wants to subscribe to a published object. The Subscribe service is sent by the potential subscriber, using the selected AR, to the publisher of the published object.

#### Published Object

This parameter specifies one of the key attributes of the published object.

#### Schedule Interval

This parameter specifies the increment in the fieldbus time between successive starting times for the scheduled sequence. The time granularity is the same as that defined for the underlying data-link layer (see IEC 61158-3-1).

#### Maximum ARs

This optional parameter specifies the maximum number of ARs on which the subscriber is willing to receive data. The default value for this parameter is one (1).

NOTE In certain circumstances, a Publisher AP may be requested to publish the same object on more than one AR. A subscriber AP may receive the same published object from a Publisher AP on more than one AR. The AR on which a Subscribe service is conducted may be different from the AR used to publish the data.

**Maximum Subscriber AP Permissible Jitter**

This optional parameter specifies the single upper bound on the two definitions of scheduling jitter on which the subscriber AP is willing to receive the data:

a) the difference in starting times of a given scheduled sequence within successive subscriber AP cycles, and,

b) the difference in starting times of a given subscriber AP cycle relative to a DLL cycle.

The value of this parameter is always less than the value of the Cycle Interval parameter. If this parameter is not specified, the jitter will be determined by the publisher. The granularity may be the same as defined for the data-link layer (see IEC 61158-3-1).

**Desired Publishing Start Time**

This optional parameter specifies the fieldbus time at which publishing should start.

**Earliest Publishing Start Time**

This optional parameter specifies the earliest fieldbus time at which publishing may start.

**Latest Publishing Start Time**

This optional parameter specifies the latest fieldbus time at which the publishing is to start.

**Leaving**

This parameter is to be selected if a subscriber wants to stop its subscription to a specific object through a given AREP. The Subscribe service is sent to the Publisher AP using the AR used for joining.

**List of Leaving AREPs**

This parameter specifies the push publisher's AREP(s) that the subscribing AP wants to leave.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Data type**

This conditional parameter identifies the data type of the variable. This parameter is present if the subscriber is attempting to join a relationship.

**Data Length**

This conditional parameter specifies the length in octets of the data value to be published. This parameter is present if the subscriber is attempting to join a relationship and if the length is not defined for the data type of the published object.

**Dedicated**

This conditional parameter, when TRUE, indicates that the publishing relationship is dedicated. This parameter is present if the subscriber is attempting to join a relationship.

**List of AR Information**

This conditional parameter specifies the information necessary for the subscriber to construct its local subscriber AREP(s). This parameter is present if the subscriber is attempting to join a relationship. If present, this list is not empty.

**Publishing AREP ID**

This parameter specifies the numeric identifier for the AREP used for publishing.

**DL Mapping Reference**

This parameter identifies the DL mapping for the publishing AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**Scheduled start time**

This parameter specifies the actual publication start time.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.2.3.2.3 Service procedure

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

If the request indicates that a subscriber wishes to "join", the publishing AP locally determines if it can accommodate the specifics of the request. If it can, it may (1) reconfigure existing AR endpoint(s), or (2) allocate resources and create a new publishing AR endpoint(s), or (3) allocate bandwidth in the network schedule if necessary.

If the request indicates that a subscriber wishes to "leave", the AP locally determines if it can cease publishing over the indicated AR and still accommodate its other existing subscribers. If so, the AP may locally free up the AREP(s) and cancel the related network schedule(s).

If a confirmation (+) was returned for a "join" operation, the FAL user in the AP receiving the response APDU may configure or may create its local AREP(s) that will be used to receive the published variable.

If a confirmation (+) was returned for a "leave" operation, the FAL user in the AP may locally free up the resources associated with the AREP(s) used for subscription

It is a local matter for the FAL user to determine if the schedule specified by the publisher is sufficient. If insufficient, the local user should issue a Subscribe service primitive to inform the publisher that it is leaving.

### 6.2.2.3.3 Identify service

#### 6.2.2.3.3.1 Service overview

Information to identify an AP is read with this service.

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

#### 6.2.2.3.3.2 Service primitives

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

**Table 18 – Identify**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Vendor Name | | | M | M (=) |
| Model Identifier | | | M | M (=) |
| Vendor Revision | | | M | M (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Vendor Name**

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

**Model Identifier**

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

**Vendor Revision**

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

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.2.3.3.3 Service procedure

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

### 6.2.2.3.4 Get status service

#### 6.2.2.3.4.1 Service overview

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

#### 6.2.2.3.4.2 Service primitives

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

**Table 19 – Get status**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Service Access Status | | | M | M (=) |
| Operational Status | | | M | M (=) |
| Local Detail | | | U | U (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Service Access Status**

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

**Operational Status**

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

**Local Detail**

This parameter specifies the user defined state of the application.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.2.3.4.3 Service procedure

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

### 6.2.2.3.5 Status notification service

#### 6.2.2.3.5.1 Service overview

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

#### 6.2.2.3.5.2 Service primitives

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

**Table 20 – Status notification**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Service Access Status | M | M (=) |
| Operational Status | M | M (=) |
| LocalDetail | 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.

**Service Access Status**

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

**Operational Status**

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

**Local Detail**

This parameter specifies the user defined state of the application.

#### 6.2.2.3.5.3 Service procedure

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

### 6.2.2.3.6 Initiate service

#### 6.2.2.3.6.1 Service overview

This service is used to establish a context between communicating APs for the exchange information on a single AR. The Initiate service negotiates the supported FAL services, the maximum outstanding services permitted, the maximum PDU length and the current version of the Object Definitions. The data structure nesting level is also negotiated if the local and remote APs support data structure nesting levels greater than one. The supported FAL services, the maximum outstanding services permitted, the maximum PDU length and data structure nesting level are AP system management attributes configured for the AREP. This service may be used to dynamically create the AREP to be used if the AREP does not exist.

#### 6.2.2.3.6.2 Service primitives

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

**Table 21 – Initiate**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Version Object Definitions Calling | M | M (=) | | |
| AP Descriptor Calling | M | M (=) | | |
| Access Protection Supported Calling | M | M (=) | | |
| Password Calling | M | M (=) | | |
| Access Groups Calling | M | M (=) | | |
| Destination DL-Address | C | | | |
| DL-Mapping | C | C | | |
| Nesting Level Calling | U | U (=) | | |
| User Detail | U | U (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Version OD Called | | | M | M (=) |
| AP Descriptor Called | | | M | M (=) |
| Access Protection Supported Called | | | M | M (=) |
| Password Called | | | M | M (=) |
| Access Groups Called | | | M | M (=) |
| DL-Mapping | | | C | C |
| Nesting Level Called | | | U | U (=) |
| User Detail | | | U | U (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Error Code | | | M | M (=) |
| Max PDU Length Sending Called | | | | C |
| Max PDU Length Receiving Called | | | | C |
| FAL Features Supported Called | | | | C |

NOTE See the Note in 3.8.4.3.

**Argument**

This parameter carries the parameters of the service invocation.

**Version Object Definitions Calling**

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

**AP Descriptor Calling**

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

**Access Protection Supported Calling**

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

**Password Calling**

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

**Access Groups Calling**

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

**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.

**DL-Mapping**

This conditional parameter specifies the DL-Mapping attributes for the AREP to be opened. It is used only when the AREP to be opened is not defined.

**Nesting Level Calling**

This optional parameter indicates the nesting level supported by the requester.

**User Detail**

This optional parameter specifies user information.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Version Object Definition Called**

This parameter should specify the version of the server's Object Definitions.

**AP Descriptor Called**

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

**Access Protection Supported Called**

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

**Password Called**

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

**Access Groups Called**

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

**DL-Mapping**

This conditional parameter specifies the negotiated DL-Mapping attributes for the AREP to be opened. It is used only when the AREP to be opened is not defined.

**Nesting Level Called**

This optional parameter indicates the nesting level supported by the responder.

**User Detail**

This optional parameter specifies user information.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Error Code**

This parameter should provide the reason for the failure.

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

**Max PDU Length Sending Called**

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

It should be transmitted by the called FAL and should only be part of the Initiate.cnf primitive.

**Max PDU Length Receiving Called**

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

It should be transmitted, if supported, by the called FAL and should only be part of the Initiate.cnf primitive.

**FAL Services Supported Called**

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

It should be transmitted, if supported, by the called FAL and should only be part of the Initiate.cnf primitive.

#### 6.2.2.3.6.3 Service procedure

This service operates through a queue.

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

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

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

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

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

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

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

#### 6.2.2.3.7 Terminate service

##### 6.2.2.3.7.1 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.

##### 6.2.2.3.7.2 Service primitives

The service parameters for this service are shown in Table 22.

**Table 22 – Terminate**

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

**Argument**

This parameter carries the parameters of the service invocation.

**Locally Generated**

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

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

**Terminate Identifier**

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

FAL USER
FAL APO ASE
FAL AR ASE
DLL

**Reason Code**

This parameter should specify the reason for the termination.

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

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

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

| Reason | Meaning |
|---|---|
| AR Error | Faulty AR |
| User Error | Improper, unknown or faulty service primitive received from the FAL user |
| FAL PDU Error | Unknown or faulty FAL PDU received from the AR ASE |
| Connection State Conflict AR ASE | Improper AR ASE service primitive |
| AR ASE Error | Unknown or faulty AR ASE service primitive |
| PDU Size | PDU length exceeds maximum PDU length |
| Feature Not Supported | SERVICE-REQ PDU received from the AR ASE and service or option not supported as a server (see FAL Features Supported attribute in the AR) |
| Unexpected service response | Unexpected confirmed service.rsp received from the FAL user, or unexpected CONFIRMED_SERVICE-RSP_PDU received from the AR ASE |
| 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. |

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

**Terminate Detail**

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

#### 6.2.2.3.7.3 Service procedure

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

### 6.2.2.3.8 Conclude service

#### 6.2.2.3.8.1 Service overview

This service should be used to gracefully release an existing AR between APs.

#### 6.2.2.3.8.2 Service primitives

The service parameters for this service are shown in Table 23.

**Table 23 – Conclude**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Result(+) | | | S | S (=) |
| Result(–) | | | S | S (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

This parameter carries the parameters of the service invocation.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.2.3.8.3 Service procedure

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

### 6.2.2.3.9 Reject service

#### 6.2.2.3.9.1 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.

#### 6.2.2.3.9.2 Service primitives

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

**Table 24 – Reject**

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

**Argument**

The argument contains the parameters of the service request.

**Reject Code**

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

Unsupported Service

AR Not Established

AR Not Defined

Max Outstanding Requests Exceeded

Max PDU Size Exceeded

**Original FAL Service Type**

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

**Original PDU Body**

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

#### 6.2.2.3.9.3 Service procedure

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

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

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

### 6.2.3 Application relationship ASE

#### 6.2.3.1 Overview

##### 6.2.3.1.1 General

In a distributed system, application processes communicate with each other by exchanging application layer messages across well-defined application layer communications channels. These communication channels are modeled in the fieldbus Application Layer as application relationships (ARs).

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

The messages that are conveyed by ARs are FAL service requests and responses. Each is submitted to the AR ASE for transfer by an FAL Application Service Element (ASE) that represents the class of the APO being accessed. Figure 1 illustrates this concept.

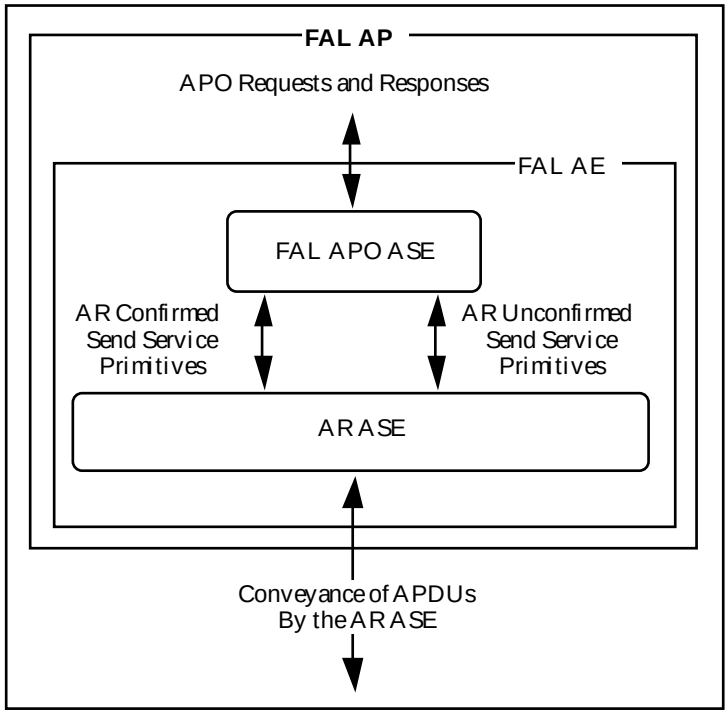



**Figure 1 – The AR ASE conveys APDUs between APs**

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

#### 6.2.3.1.2 Endpoint context

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

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

##### 6.2.3.1.2.1 Endpoint role

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

NOTE See concepts subclause for description of clients and servers, and the push and pull models for publishers and subscribers.

Table 25 and Table 26 summarize the characteristics and combinations of each of the AREP roles.

**Table 25 – Conveyance of service primitives by AREP role**

| | Client | Server | Peer | Push Publisher | Push Subscriber | Pull Publ Mgr | Pull Publisher | Pull Subscriber |
|---|---|---|---|---|---|---|---|---|
| Send Service Req | X | | X | X | | X | | |
| Recv Service Req | | X | X | | X | | X | |
| Send Service Rsp | | X | X | | | | X | |
| Recv Service Rsp | X | | X | | | X | | X |

**Table 26 – Valid combinations of AREP roles involved in an AR**

| AREP Roles by Interaction Model | Client | Server | Peer | Push Publisher | Push Subscriber | Pull Publ Mgr | Pull Publisher | Pull Subscriber |
|---|---|---|---|---|---|---|---|---|
| Client/server | | | | | | | | |
| Client | | Yes | Yes | | | | | |
| Server | | | Yes | | | | | |
| Peer | | | Yes | | | | | |
| Publisher/subscriber | | | | | | | | |
| Push Publisher | | | | | Yes | | | |
| Push Subscriber | | | | | | | | |
| Pull Publ Mgr | | | | | | | Yes | |
| Pull Publisher | | | | | | | | Yes |
| Pull Subscriber | | | | | | | | |

#### 6.2.3.1.2.2 Dedicated AR endpoints

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

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

#### 6.2.3.1.2.3 Cardinality

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

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

When expressed from the viewpoint of a publisher endpoint, they indicate that multiple subscribers are supported. These ARs are 1-to-many. ARs that are 1-to-many provide for communications between one application and a group of (one or more) applications. They are often referred to as multi-party and multi-cast. To support the “pull” model of publishing, these ARs provide a management path between the publishing manager and the publisher.

In 1-to-many ARs, the publisher endpoint identifies the remote endpoints using a group identifier, rather than identifying each one individually as is done with 1-to-1 ARs. This permits subscribers to join and leave ARs without disrupting the publisher endpoint context because their individual identities are not known to the publisher endpoint. However, the publisher application may be aware of their participation, but that information is not part of the AR endpoint context.

#### 6.2.3.1.2.4 Conveyance model

##### 6.2.3.1.2.4.1 General

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

conveyance paths

trigger policy, and

conveyance policy.

##### 6.2.3.1.2.4.2 Conveyance paths

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

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

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

##### 6.2.3.1.2.4.3 Trigger policy

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

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

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

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

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

##### 6.2.3.1.2.4.4 Conveyance policy

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

Buffered ARs contain conveyance paths that have a single buffer at each endpoint. Updates to the source buffer are conveyed to the destination buffer according to the trigger policy of the AR. Updates to either buffer replace its contents with new data. In buffered ARs, unconveyed or undelivered data that has been replaced is lost. Additionally, data contained in a buffer may be read multiple times without destroying its contents.

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

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

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

#### 6.2.3.1.3 Underlying communications services

##### 6.2.3.1.3.1 General

The AR ASE conveys FAL APDUs using the capabilities of the underlying data-link layer. Several characteristics are used to describe these capabilities. This subclause provides a description of each. These characteristics are specific to the data-link Mapping defined in IEC 61158-6-5. Their precise specification can be found there.

##### 6.2.3.1.3.2 Connectionless and connection-oriented services

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

NOTE Connectionless and connection-oriented are DL terms. Refer to IEC 61158-3-1 and IEC 61158-4-1 for definition.

##### 6.2.3.1.3.3 Buffered and queued services

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

##### 6.2.3.1.3.4 Cyclic and acyclic transfers

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

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

##### 6.2.3.1.3.5 Quality of service

The underlying layer may provide more than one quality of service. When this occurs, the AREP should be configured to use the quality of service appropriate for the APDUs it conveys. Quality of service parameters for AREPs are contained in the data link mappings in IEC 61158-6-5.

##### 6.2.3.1.3.6 Segmentation

The FAL has the capability of segmenting user data conveyed by it. In addition, when the underlying layer provides segmentation, too, the maximum length of data conveyed by the FAL in a single APDU will increase accordingly.

#### 6.2.3.1.3.7 Address format

The underlying layer may support more than one address format. When it does, all endpoints of an AR should be mapped to compatible address formats. DL-mappings for AREPs are specified in IEC 61158-6-5 for the data-link layer (IEC 61158-3-1).

#### 6.2.3.1.3.8 Timeliness

AR endpoints may be defined to support timeliness information about their transfers as calculated by the data-link layer. This timeliness information is received from the data-link layer and forwarded to the FAL user as a parameter of the associated FAL service. Publisher and subscriber AR endpoints support transparent, residence, synchronized, and update timeliness. These types of timeliness are described in the definitions of each of these endpoint types below. The mechanism used by the data-link layer to calculate timeliness of published data is defined in IEC 61158-3-1 and IEC 61158-4-1. The attributes defined by the AR Model to support timeliness are defined in the DLL Mapping Attributes for the Publisher/Subscriber AREPs in IEC 61158-6-5.

#### 6.2.3.1.3.9 Duplicate FAL PDU detection

AR endpoints may be defined to support detection of the receipt of duplicate FAL APDUs. Alternatively, an indication may be received from the data-link layer to support this capability and forwarded to the FAL user as a parameter of the associated FAL service. The attributes defined by the AR Model to support this capability are defined in the DLL Mapping Attributes in IEC 61158-6-5.

### 6.2.3.1.4 AR establishment

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

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

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

Third, ARs can require dynamic definition and establishment. In this case, definitions have to be created for each of the AREPs using the Create service. After creation, they are established using the Establish service if they were not created as “established”.

### 6.2.3.1.5 Application relationship classes

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

## 6.2.3.2 Application relationship endpoint class specifications

### 6.2.3.2.1 Formal model

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

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

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | AR ENDPOINT | | |
| **CLASS ID:** | | | 32 |
| **PARENT CLASS:** | | | TOP |
| **SYSTEM MANAGEMENT ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Local AP |
| 2 | (m) | Attribute: | FAL Revision |
| 3 | (m) | Attribute: | Dedicated (TRUE, FALSE) |
| 4 | (o) | Attribute: | Transfer Syntax |
| 5 | (o) | Attribute: | List Of Supported Attributes |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | AR-Unconfirmed Send |
| 2 | (o) | OpsService: | AR-Confirmed Send |
| 3 | (o) | OpsService: | AR-Establish |
| 4 | (o) | OpsService: | AR-DeEstablish |
| 5 | (o) | OpsService: | AR-Abort |
| 6 | (o) | OpsService: | AR-Compel |
| 7 | (o) | OpsService: | AR-Get Buffered Message |
| 8 | (o) | OpsService: | AR-Schedule Communication |
| 9 | (o) | OpsService: | AR-Cancel Scheduled Sequence |
| 10 | (o) | OpsService: | AR-Get DL-Time |
| 11 | (o) | OpsService: | AR-Status |
| 12 | (o) | OpsService: | AR-XON-OFF |
| 13 | (o) | OpsService: | AR-RemoteRead |
| 14 | (o) | OpsService: | AR-RemoteWrite |

#### 6.2.3.2.2 System management attributes

**Local AP**

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

**FAL Revision**

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

**Dedicated**

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

**Transfer Syntax**

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

**List of Supported Attributes**

This optional attribute specifies the attributes supported by the object. This list contains, at a minimum, the mandatory attributes for the class of the object.

#### 6.2.3.2.3 Services

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

**AR-Unconfirmed Send**

This optional service is used to send an unconfirmed service.

**AR-Confirmed Send**

This optional service is used to send a confirmed service.

**AR-Establish**

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

**AR-DeEstablish**

This optional service is used by client or peer users to request the graceful termination of a 1-to-1 AR.

**AR-Abort**

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

**AR-Compel**

This optional service is used by the FAL user to request the AR ASE to convey a message which has been deferred until explicitly released.

**AR-Get Buffered Message**

This optional service is used to request the AR to retrieve a message which is being maintained in a buffer in the local data-link layer.

**AR-Schedule Communication**

This optional service is used to to schedule a sequence of DL-COMPELs for a particular relationship.

**AR-Cancel Scheduled Sequence**

This optional service is used to cancel an existing sequence which has previously been scheduled.

**AR-Get DL-Time**

This optional service is used to access the DL-Time service of the data-link layer.

**AR-Status**

This optional service is used to indicate to the AR user that an internal event of the AREP, such as the transmission of a network-triggered buffer, has occurred.

**AR-XON-OFF**

This optional service causes the flow of data on a specified AR to be suspended or resumed.

**AR-RemoteRead**

This optional service is used to request the AR to retrieve a message which is being maintained in a buffer in the remote data-link layer.

**AR-RemoteWrite**

This optional service is used to to trigger a data exchange, after writing a value in a local buffer.

#### 6.2.3.3 Application relationship ASE service specifications

##### 6.2.3.3.1 Supported services

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

AR-Unconfirmed Send

AR-Confirmed Send

AR-Establish

AR-DeEstablish

AR-Abort

AR-Compel

AR-Get Buffered Message

AR-Schedule Communication

AR-Cancel Scheduled Sequence

AR-Status

AR-XON-OFF

AR-RemoteRead

AR-RemoteWrite

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

#### 6.2.3.3.2 AR-unconfirmed send service

##### 6.2.3.3.2.1 Service overview

This service is used to send AR-Unconfirmed request APDUs for FAL APO ASEs. The AR-Unconfirmed Send service may be requested at either endpoint of a one-to-one bi-directional AR, at the server endpoint of a one-to-one uni-directional AR, or at the publisher endpoint of a 1-to-many AR.

NOTE This service is described abstractly in such a way that it is capable of operating with ARs that convey FAL APDUs through buffers or queues. This service may be implemented in such a way that the capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying data-link layer. Alternatively, this service may be implemented such that these capabilities are combined so that the user may load the buffer/queue and request its transfer in a single operation.

##### 6.2.3.3.2.2 Service primitives

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

**Table 27 – AR-Unconfirmed send**

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

**Argument**

The argument contains the parameters of the service request.

**Remote DLSAP Address**

This conditional parameter specifies the destination DLSAP address in the request and the source DLSAP address in the indication. It is present if the AREP is configured with a Remote Address Configuration Type of FREE.

**FAL Service Type**

This parameter specifies the type of the service being conveyed.

**FAL APDU Body**

This parameter specifies the service dependent body for the APDU.

**Timeliness**

This conditional parameter indicates both the local and remote timeliness status of the FAL APDU Body contained in the AR-Unconfirmed Send request primitive. This parameter is present if Timeliness is supported in the DL Mapping Attributes of the AREP. The values associated with this parameter are defined in the description of the DLL Mapping Attributes in IEC 61158-6-5.

**Duplicate FAL PDU Body**

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the data-link layer. It is present if supported in the DL Mapping Attributes of the AREP.

#### 6.2.3.3.2.3 Service procedure

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

The requesting FAL ASE submits an AR-Unconfirmed send.request primitive to its AR ASE. The AR ASE builds an AR-Unconfirmed Send request APDU. If the local AREP for the specified AR supports timeliness, the AR ASE indicates the publishing and transmission timeliness in the APDU.

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

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

NOTE The transmission schedule is managed by the underlying layer, not the AR-ASE. Refer to IEC 61158-3-1 and IEC 61158-4-1 for further details.

Upon receipt of the AR-Unconfirmed Send request APDU, the receiving AR ASE delivers an AR-Unconfirmed send.indication primitive to the appropriate FAL ASE as indicated by the FAL Service Type Parameter. If the receiving AREP supports timeliness, the AR ASE includes the timeliness parameters received from the data-link layer in the indication primitive.

### 6.2.3.3.3 AR-confirmed send service

#### 6.2.3.3.3.1 Service overview

This service is used to send confirmed request and response APDUs for FAL APO ASEs. The AR-Confirmed Send service may be requested at Client and Peer endpoints of one-to-one bi-directional ARs.

NOTE This service is described abstractly in such a way that it is capable of operating with ARs that convey FAL APDUs through buffers or queues. This service may be implemented in such a way that the capability is provided

to load the buffer/queue, and subsequently post it for transfer by the underlying data-link layer. Alternatively, this service may be implemented such that these capabilities are combined so that the user may load the buffer/queue and request its transfer in a single operation.

#### 6.2.3.3.3.2 Service primitives

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

**Table 28 – AR-Confirmed send**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Destination DL-Address | C | C | | |
| FAL Service Type | M | M (=) | | |
| FAL APDU Body | M | M (=) | | |
| Result | | | M | M (=) |
| Invoke ID | | | | U (=) |
| Source DL-Address | | | C | C |
| FAL Service Type | | | M | M (=) |
| FAL APDU Body | | | M | M (=) |
| Timeliness | | | | C |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Destination DL-Address**

This conditional parameter specifies the Destination DL-address. It is present only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE.

NOTE 1 Not all confirmed services support this parameter. The use of this parameter when using confirmed services is a local matter.

**FAL Service Type**

This parameter specifies the type of the service being conveyed.

**FAL APDU Body**

This parameter specifies the service dependent body for the APDU.

**Result**

This selection type parameter indicates that the service request succeeded.

**Source DL-Address**

This conditional parameter specifies the Source DL-address. It is present only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE.

NOTE 2 Not all confirmed services support this parameter. The use of this parameter with confirmed services is a local matter.

**FAL Service Type**

This parameter specifies the type of the service being conveyed.

**FAL APDU Body**

This parameter specifies the service dependent body for the APDU.

**Timeliness**

This conditional parameter indicates the timeliness of the update into the FAL. This parameter is present if Timeliness is supported in the DL Mapping Attributes of the AREP. The values associated with this parameter are defined in the description of the DLL Mapping Attributes in IEC 61158-6-5.

#### 6.2.3.3.3.3 Service procedure

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

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

The AR ASE builds an AR-Confirmed Send request APDU. If the local AREP for the specified AR supports timeliness, the AR ASE indicates the publishing and transmission timeliness in the APDU.

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

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

NOTE The transmission schedule is managed by the underlying layer, not the AR-ASE. Refer to IEC 61158-3-1 and IEC 61158-4-1 for further details.

Upon receipt of the AR-Confirmed Send request APDU, the receiving AR ASE delivers an AR-Confirmed Send indication primitive to the appropriate FAL ASE as indicated by the FAL Service Type Parameter. If the receiving AREP supports timeliness, the AR ASE includes the timeliness parameters received from the data-link layer in the indication primitive.

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

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

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

Upon receipt of the confirmed send response APDU, the receiving AR ASE uses identifying information contained in the response APDU to associate the response with the appropriate request and cancel the associated transaction state machine. The AR ASE delivers an AR-Confirmed Send confirmation primitive to the requesting FAL ASE. If the receiving AREP supports timeliness, the AR ASE includes the timeliness parameters received from the data-link layer in the confirmation primitive.

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

#### 6.2.3.3.4 AR-establish service

##### 6.2.3.3.4.1 Service overview

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

The endpoint context may be created during establishment or prior to establishment through System Management. Once defined, the Set Attributes and Get Attributes services can be used to update and further coordinate endpoint contexts.

##### 6.2.3.3.4.2 Service primitives

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

**Table 29 – AR-Establish service**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Remote DL-address | C | C (=) | | |
| User Data | U | U (=) | | |
| Result | | | M | M (=) |
| User Data | | | U | U (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**AREP**

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

**Remote DL-address**

This conditional parameter identifies the remote DL-address. If the AR to be established is supported by a DL-Connection, then the DL-address is a DLCEP-address. If not, the DL-address is a DLSAP-address. It is present when the AREP Remote Address Configuration Type attribute is FREE.

**User Data**

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

**Result**

This parameter indicates that the service request succeeded or failed.

**AREP**

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

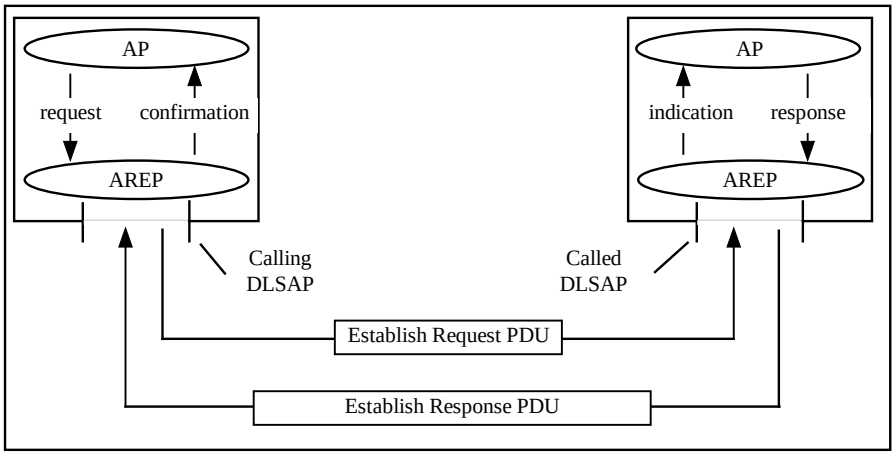
**User Data**

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

#### 6.2.3.3.4.3 Service procedure

##### 6.2.3.3.4.3.1 1-to-1 AR establishment

When used in support of 1-to-1 AREPs, the AR-Establish service causes Establish PDUs to be exchanged between calling and called DLSAPs as shown in Figure 2.




The Establish Request PDU is sent from the calling DLSAP AP to a called DLSAP.

The APDU contains identifiers for the AREPs to be related.

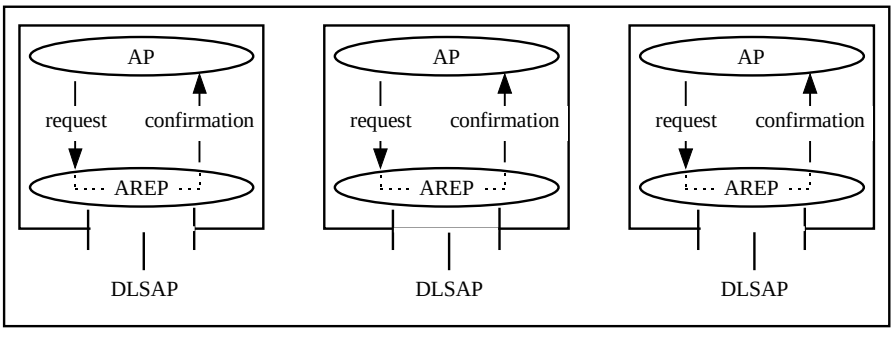
If the called AP processes the establishment request, its AREP returns an Establish Response PDU.

**Figure 2 – 1-to-1 AR establishment**

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

##### 6.2.3.3.4.3.2 1-to-many AR establishment

When used in support of 1-to-Many AREPs, the AR-Establish service causes local AREPs to be established independently, as shown in Figure 3.




Each AP in the AR establishes its own AREP locally by issuing an Establish request. Establish PDUs are not exchanged.

However, if the AR is supported by a data link connection, DL-PDUs are exchanged

**Figure 3 – 1-to-many AR establishment**

Upon receipt of an AR-Establish request service primitive, the calling FAL issues an AR-Establish confirmation service primitive to the calling AP indicating whether or not it was able to establish the AREP. If one of the following cases is found, it was not able to establish the AREP:

1) the requested AREP is not specified within the FAL, or
2) resources are not available to establish the requested AREP, or

3) for AREPs supported by a data link connection, the data-link layer was not able to establish the data link connection.

#### 6.2.3.3.4.3.3 Compatible AREP classes

Table 30 provides possible combinations of the AREP classes which may be related with the AR-Establish service. In this table the SERVER column contains a "-" to indicate that Server AREPs do not initiate AR establishment.

**Table 30 – Valid combinations of AREP classes to be related**

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

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

#### 6.2.3.3.4.3.4 Conflict resolution

The normal establishment of ARs follows the general time-sequence for confirmed services. In the cases where each endpoint of an AR concurrently issues an AR-Establish request, a conflict arises. The algorithms for resolving conflicts of this type are specified in detail in IEC 61158-6-5.

### 6.2.3.3.5 AR-deEstablish service

#### 6.2.3.3.5.1 Service overview

This confirmed service is invoked by client or peer users to request the graceful termination of a 1-to-1 application relationship. Use of the AR-DeEstablish service causes the endpoints of the AR to be closed. They may be reopened using the AR-Establish service. This service may be used on any open AREP, whether or not the AR was pre-established or dynamically established using the AR-Establish service.

When a request to de-establish an AR is made, the local AR ASE conveys the DeEstablish Request PDU to the remote endpoint of the AR, and accepts no additional request PDUs from the user unless an AR-Abort is requested. Upon receipt of a DeEstablish Response PDU, it clears all outstanding service requests, closes the endpoint context, and informs the user using the confirm service primitive.

The remote AR ASE informs its user when an AR-DeEstablish Request PDU is received using the indication primitive for the service. The user returns a response primitive after responding to the service indications it received. When the remote AR ASE receives the response, it closes the endpoint context, and returns an AR- DeEstablish Response PDU.

#### 6.2.3.3.5.2 Service primitives

The service parameters for this service are shown in Table 31. The use of this service is for future study.

**Table 31 – AR-Deestablish service**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**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.

#### 6.2.3.3.5.3 Service procedure

The AR-DeEstablish Service is a service that operates through a queue or buffer. Its service procedure is for future study.

### 6.2.3.3.6 AR-abort service

#### 6.2.3.3.6.1 Service overview

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

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

The AR-Abort service may be requested at either AREP of a one-to-one relationship, or at the publisher endpoint of a 1-to-many or 1-to-all AR. Subscribers are not capable of aborting ARs with publishers, although they may close their own endpoints through local means.

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

#### 6.2.3.3.6.2 Service primitives

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

**Table 32 – AR-Abort**

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

**Argument**

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

**Locally Generated**

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

**Originator**

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

**Reason Code**

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

**Additional Detail**

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

#### 6.2.3.3.6.3 Service procedure

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

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

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

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

### 6.2.3.3.7 AR-compel service

#### 6.2.3.3.7.1 Service overview

This service is used by the FAL user to request the AR ASE to convey a message which has been deferred until explicitly released.

NOTE The services that operate on ARs are described abstractly in such a way that they are capable of operating with ARs that convey FAL APDUs through buffers or queues. These services may be implemented such that the

capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying data-link layer using this service.

#### 6.2.3.3.7.2 Service primitives

The service parameters for this service are shown in Table 33.

**Table 33 – AR-Compel service**

| Parameter name | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Schedule ID | U | |
| Result (+) | | S |
| Status | | M |
| Result (-) | | S |
| Error Info | | M |
| NOTE See the Note in 3.8.4.3. | | |

**Argument**

The argument contains the parameters of the service request.

**Schedule ID**

This optional parameter specifies the data-link layer sequence to be compelled for network scheduled ARs. The scheduling sequence is part of the DL-Mapping defined in IEC 61158-6-5.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**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.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.3.3.7.3 Service procedure

The AR-Compel service is a service that operates through a queue or buffer.

The requesting user submits an AR compel.request primitive to its FAL AR ASE. The FAL AR ASE issues the corresponding data-link layer service request to the data-link layer.

### 6.2.3.3.8 AR-get buffered message service

#### 6.2.3.3.8.1 Service overview

This local service is used by an application process to request the AR ASE to retrieve a message which is being maintained in a buffer in the local data-link layer.

This service does not result in the conveyance of an APDU. It is provided so that the FAL user may access a buffer through the FAL AR.

#### 6.2.3.3.8.2 Service primitives

The service parameters for this service are shown in Table 34.

**Table 34 – AR-Get buffered message service**

| Parameter name | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Result(+) | | S |
| Decoded buffer data | | M |
| Local timeliness | | C |
| Remote timeliness | | C |
| Duplicate FAL PDU Body | | C |
| Result (-) | | S |
| Error Info | | M |
| NOTE See the Note in 3.8.4.3. | | |

**Argument**

The argument contains the parameters of the service request.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Decoded Buffer Data**

This parameter specifies the user data in the FAL APDU read from the buffer.

**Local Timeliness**

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data 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. It is present if supported in the DL Mapping Attributes of the AREP.

**Remote Timeliness**

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data 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. It is present if supported in the DL Mapping Attributes of the AREP.

**Duplicate FAL PDU Body**

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the data-link layer. It is present if supported in the DL Mapping Attributes of the AREP.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.3.3.8.3 Service procedure

This service requests the FAL to return the current contents of the buffer in a confirmation(+) primitive. If the buffer is empty, a confirmation(-) primitive is returned.

### 6.2.3.3.9 AR-schedule communication service

#### 6.2.3.3.9.1 Service overview

This local service provides the AL user with the ability to schedule a sequence of DL-COMPELs for a particular relationship. It maps directly to the DL-Schedule-Sequence service and has no effect on the AR state machines.

This service does not result in the conveyance of an APDU. It may, however, result in the transfer of data-link layer PDUs.

#### 6.2.3.3.9.2 Service primitives

The service parameters for this service are shown in Table 35.

**Table 35 – AR-Schedule communication service**

| Parameter name | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Invoke ID | U | |
| Schedule Information | M | |
| | | |
| Result (+) | | S |
| Invoke ID | | U (=) |
| Sequence ID | | M |
| | | |
| Result (-) | | S |
| Invoke ID | | U (=) |
| Error Info | | M |
| NOTE See the Note in 3.8.4.3. | | |

**Argument**

The argument contains the parameters of the service request.

**Schedule Info**

This specifies the data-link layer scheduling information for the scheduled communication.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Sequence ID**

This parameter provides a short-hand identifier for the requested scheduled communication.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.3.3.9.3 Service procedure

This service requests the communication stack to create a scheduled sequence.

#### 6.2.3.3.10 AR-cancel scheduled sequence service

##### 6.2.3.3.10.1 Service overview

This local service provides the AL user with the ability to cancel an existing sequence which has previously been scheduled. It maps directly to the DL-Cancel-Schedule service and has no effect on the AR state machines.

This service does not result in the conveyance of an APDU. It may, however, result in the conveyance of data-link layer PDUs.

##### 6.2.3.3.10.2 Service primitives

The service parameters for this service are shown in Table 36.

**Table 36 – AR-Cancel scheduled sequence service**

| Parameter name | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Invoke ID | U | |
| Sequence ID | M | |
| | | |
| Result (+) | | S |
| Invoke ID | | U (=) |
| | | |
| Result (-) | | S |
| Invoke ID | | U (=) |
| Error Info | | M |
| NOTE See the Note in 3.8.4.3. | | |

**Argument**

The argument contains the parameters of the service request.

**Sequence ID**

This parameter specifies the schedule of the sequence to be cancelled.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

##### 6.2.3.3.10.3 Service procedure

This service requests the communication stack to cancel an existing scheduled sequence.

#### 6.2.3.3.11 AR-get DL-time service

##### 6.2.3.3.11.1 Description

This service is defined for the FAL to provide the FAL User with access to the DL-Time service of the data-link layer. To maintain compatibility with the DL-Time service, the service and parameters definitions are not defined in this standard. See IEC 61158-3-1 for the definition of the DL-Time service.

#### 6.2.3.3.12 AR-status service

##### 6.2.3.3.12.1 Service overview

This local service provides the AL user notification of a status change of the AREP.

##### 6.2.3.3.12.2 Service primitives

The service parameters for this service are shown in Table 37.

**Table 37 – AR-Status**

| Parameter name | Ind |
|---|---|
| Argument | |
| AREP | M |
| Status code | M |

**Argument**

This parameter carries the parameters of the service invocation.

**Status Code**

This specifies the status change being reported. The following status codes are defined

lower layer reset;

buffer received;

buffer transmitted;

transmission not timely;

lower layer lost schedule – rescheduling required;

local confirmation.

##### 6.2.3.3.12.3 Service procedure

This service indicates that a significant event, as defined by the status code parameter, occurred in the communication stack.

#### 6.2.3.3.13 AR-XON-OFF service

##### 6.2.3.3.13.1 Service overview

This local service causes the flow of data on a specified AR to be suspended or resumed.

##### 6.2.3.3.13.2 Service primitives

The service parameters for this service are shown in Table 38.

**Table 38 – AR-XON-OFF**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| XON-OFF | M | M (=) |

**Argument**

The argument contains the parameters of the service request.

**XON-OFF**

This parameter specifies user supplied data that can be conveyed with the service request. User data can be "ON“ or "OFF“.

#### 6.2.3.3.13.3 Service procedure

The AR-XON-OFF Service is a service that operates through a queue.

The requesting FAL ASE submits an AR-XON-OFF.request primitive to its AR ASE. The AR ASE builds an AR-XON-OFF request APDU.

NOTE The transmission schedule is managed by the underlying layer, not the AR-ASE. Refer to IEC 61158-3-1 and IEC 61158-4-1 for further details.

Upon receipt of the AR-XON-OFF request APDU, the receiving AR ASE delivers an AR-XON-OFF.indication primitive to the appropriate FAL ASE as indicated by the FAL Service Type Parameter.

### 6.2.3.3.14 AR remote read service

#### 6.2.3.3.14.1 Service overview

This service provides the application process to request the AR ASE to retrieve a message which is being maintained in a buffer in the remote data-link layer.

#### 6.2.3.3.14.2 Service primitives

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

**Table 39 – AR-Remote read service**

| Parameter name | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Priority | M | |
| | | |
| Result (+) | | S |
| Decoded Buffer Data | | M |
| Local Timeliness | | C |
| Remote Timeliness | | C |
| | | |
| Result (-) | | S |
| Error info | | M |
| NOTE See the Note in 3.8.4.3. | | |

**Argument**

The argument contains the parameters of the service request.

**Priority**

This argument is locally used and defined by the user to allow two data flows

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Decoded Buffer Data**

This parameter specifies the user data in the FAL APDU read from the buffer.

**Local timeliness**

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data 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. It is present if supported in the DL Mapping Attributes of the AREP.

**Remote timeliness**

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data 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. It is present if supported in the the DL Mapping Attributes of the AREP.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.3.3.14.3 Service procedure

This service requests the FAL to return the current contents of the remote buffer and give the value of the contents in a confirmation(+) primitive. If the buffer is empty, a confirmation(-) primitive is returned.

### 6.2.3.3.15 AR-remote write service

#### 6.2.3.3.15.1 Service overview

This service provides the AL user with the ability to trigger a data exchange, after writing a value in a local buffer.

#### 6.2.3.3.15.2 Service primitives

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

**Table 40 – AR-Remote write service**

| Parameter name | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Priority | M | |
| Decoded Buffer Data | M | |
| Result (+) | | S |
| Result (-) | | S |
| Error Info | | M |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Priority**

This argument is locally used and defined by the user to allow two data flows.

**Decoded Buffer Data**

This argument is used to contain the APDU to be written.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.3.4 Service procedure

This service requests the FAL to write the contents of the local buffer and to trigger the data transfer. A confirmation(+) primitive is sent when the service succeeds. If the service fails, a confirmation(-) primitive is returned.

### 6.2.4 Variable ASE

#### 6.2.4.1 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.

Two classes of variable APOs are defined by the Variable Model, individual variables and variable lists. Individual variables are used to specify access to application data of any FAL-defined or user-defined data type. Services defined to access individual variables can be used to access one or more entire variables or one or more entries (per variable) in an array or one or more fields (per variable) in a structure.

Variable lists are used to group variables for access purposes. The same services used to access individual variables can be used to access a variable list.

Two types of operation are supported by variable services, “best effort” and “atomic”. The “best effort” services succeed when at least one of the referenced values can be accessed. The “atomic” services succeed when all of the referenced values can be accessed, and fail if any one of the values cannot be accessed.

When supported by the appropriate type of application relationship, the Variable Model service can be used to support two different access models, the client/server model and the publisher/subscriber model. The client/server model is characterized by a client application sending a read or write request to a server application that responds accordingly. The server's activity is stimulated by clients on the network; if there are no requests, the server generates no responses.

The publisher/subscriber model is different. It is characterized by a data producer publishing its data onto the network. Subscribers wishing to acquire the published data join the application relationship used to publish it and listen for the data as it is transmitted.

Two models are provided to support this publisher/subscriber activity, the "pull" and the "push". In the "pull" model, the publishing manager pulls the data from the publisher by issuing a read request to it. The publisher responds by multicasting a sequence of read responses to the publishing manager and to the subscribers.

In the "push" model, the publishing manager is always co-located with the publisher. Subscribers are able to indicate to the publishing manager how they want the data published. Through local interfaces, the publishing manager controls the activity of the publisher and the characteristics of the application relationships used to distribute the data. In this case, the published data is transmitted onto the network using the Information Report unconfirmed service.

Subscribers in both models receive the published data through a local copy of the data maintained by the FAL. As the data is received from the network, the local copy is updated and made available to the subscriber. When the subscriber wishes to access the data, it accesses the local copy, instead of issuing a read request to the remote variable as would happen in the client/server model. If the appropriate AREP attributes are set, timeliness information will be included with the data.

The FAL AR ASE supports both publisher/subscriber models by multi-casting the published data on buffered, network scheduled application relationships. The exact characteristics of the transfers are controlled by the attribute settings of the application relationship.

The formal model of the variable model is presented next, followed by a description of its services. IEC 61158-6-5 describes the abstract syntax and procedures for its protocol.

#### 6.2.4.2 Requirements for access to variables

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

As the Data type Model permits nesting of data structures and arrays to more than one level, the services defined for variable access permit partial access to any nesting level. Partial access means independent access to several components of each structure or array referenced. That is, if constructed type "A" contains constructed type "B", then partial access is provided to "B", and also to components of "B".

#### 6.2.4.3 Variable model class specification

##### 6.2.4.3.1 Simple variable class specification

###### 6.2.4.3.1.1 Simple variable formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **VARIABLE ASE** |
| **CLASS:** | | **SIMPLE VARIABLE** | |
| **CLASS ID:** | | | 7 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (o) | Key Attribute: | Symbolic Address |
| 2 | (m) | Attribute: | Data type |
| 3 | (m) | Attribute: | Length |
| 4 | (c) | Constraint: | Data type Format = STRING |
| 4.1 | (o ) | Attribute: | Variable Length Conveyance (TRUE, FALSE) -- see Note below |
| 5 | (m) | Attribute: | Access Privilege |
| 5.1 | (m) | Attribute: | Password |
| 5.2 | (m) | Attribute: | Access Groups |
| 5.3 | (m) | Attribute: | Access Rights |
| 6 | (m) | Attribute: | Local Detail |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Information Report |
| 6 | (o) | OpsService: | Information Report List |
| 7 | (o) | OpsService: | Exchange |
| 8 | (o) | OpsService: | Exchange List |

NOTE The constraint is TRUE when a variable is being defined as an array.

#### 6.2.4.3.1.2 Attributes

**Symbolic Address**

This optional key attribute specifies a symbolic reference for the variable. This attribute provides a second name space for defining variables that is separate from that of the variable name to ensure that duplication of names in separate name spaces do not create a problem. The symbolic address may be used to assign variable names independent of how they are named within a given system.

A space (" ") character may not be embedded in the middle of a symbolic address. However, using it as leading or trailing fillers is permitted. Such fillers are not to be interpreted as part of the symbolic address. The use of leading or trailing fillers permits fixed-length character string usage.

NOTE For example, a device manufacturer may use the symbolic address to name the data acquired from a local input channel as "port 1". A user of that data may create a second variable for the data using the variable name "temperature".

**Data type**

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

**Length**

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

**Variable Length Conveyance**

This conditional Boolean attribute, when TRUE, indicates that only the valid octets of the string are conveyed. When FALSE, all octets of the string are conveyed, even if they are not part of the string value. This attribute is present only for variables with the STRING data type.

**Access Privilege**

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

**Password**

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

**Access Groups**

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

**Access Rights**

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

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

**Local Detail**

This attribute specifies local information.

#### 6.2.4.3.1.3 Services

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

**Read**

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

**Write**

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

**Read List**

This optional service may be used to read multiple variable objects or components of multiple variable objects or a single variable list object. The number of referenced variables/components or variable objects in the list is a device constructor matter. Each of the referenced variables/components or variables in the list is accessed and returned in a “best effort” fashion, such that at least one value is to be returned for the service to succeed. This service may be used only in the client/server model.

**Write List**

This optional service may be used to update multiple variable objects or single components of multiple variable objects or a single variable list object. The number of referenced variables/components or variable objects in the list is a device constructor matter. Each of the referenced variables/components or variables in the list are updated in a “best effort” fashion, such that at least one value is to be updated for the service to succeed. This service may be used only in the client/server model.

**Information Report**

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

**Information Report List**

This optional unconfirmed service may be used to report multiple variable objects or single components of multiple variable objects or a single variable list object. The number of referenced variables/components or variable objects in the list is a device constructor matter. This service may be used in both the client/server model and the push publisher/subscriber model.

**Exchange**

This optional service is a confirmed service that may be used to write the value of a remote variable or variable list and read the value of another variable or variable list in one operation. This service may be used in the client/server model. The relationship between the specified variables and User Layer Exchange Function is outside the scope of this standard.

**Exchange List**

This optional confirmed service may be used to update multiple variable objects or single components of multiple variable objects or a single variable list object and to read multiple variable objects or single components of multiple variable objects or a single variable list object in one operation. The number of referenced variables/components or variable objects in the list is a device constructor matter. Each of the referenced variables/components or variables in the list is updated/accessed in a “best effort” fashion, such that at least one value is to be updated/returned for the service to succeed. This service may be used only in the client/server model. The relationship between the specified variables and a User Layer Exchange Function is outside the scope of this standard.

#### 6.2.4.3.2 Array variable class specification

##### 6.2.4.3.2.1 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **VARIABLE ASE** |
| **CLASS:** | | **ARRAY VARIABLE** | |
| **CLASS ID:** | | | 8 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (o) | Key Attribute: | Symbolic Address |
| 2 | (m) | Attribute: | Data type |
| 2.1 | (s) | Attribute: | Data type ID |
| 2.2 | (s) | Attribute: | Embedded Data type |
| 3 | (c) | Constraint: | Data type Format = ARRAY \| FIXED-LENGTH \| STRING |
| 3.1 | (o ) | Attribute: | Element Length |
| 4 | (c) | Constraint: | Data type Format = STRING |
| 4.1 | (o ) | Attribute: | Variable Length Conveyance (TRUE, FALSE) |
| 5 | (o ) | Attribute: | Number of Elements |
| 6 | (m) | Attribute: | Access Privilege |
| 6.1 | (m) | Attribute: | Password |
| 6.2 | (m) | Attribute: | Access Groups |
| 6.3 | (m) | Attribute: | Access Rights |
| 7 | (m) | Attribute: | Local Detail |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Information Report |
| 6 | (o) | OpsService: | Information Report List |
| 7 | (o) | OpsService: | Exchange |
| 8 | (o) | OpsService: | Exchange List |

##### 6.2.4.3.2.2 Attributes

**Symbolic Address**

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

**Data type**

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

**Data type ID**

This attribute is the numeric identifier of a data type associated with the array. If the format of the data type is ARRAY, this attribute is the data type of the array. If the data type is FIXED-LENGTH or STRING, this attribute is the data type of an array element.

**Embedded Data type**

This attribute is used to embed the data type description in the array definition. The contents for this attribute are shown in the Data type model.

**Element Length**

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

**Variable-length Conveyance**

This conditional Boolean attribute, when TRUE, indicates that only the valid octets of the string are conveyed. When FALSE, all octets of the string are conveyed, even if they are not part of the string value. This attribute is present only for arrays with a STRING data type.

**Number of Elements**

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

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

**Access Privilege**

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

**Local Detail**

This attribute specifies local information.

#### 6.2.4.3.2.3 Services

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

### 6.2.4.3.3 Record variable class specification

#### 6.2.4.3.3.1 Formal model

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

| | | | |
|---|---|---|---|
| 1 | (o) | Key Attribute: | Symbolic Address |
| 2 | (m) | Attribute: | Data type |
| 2.1 | (s) | Attribute: | Data type ID |
| 2.2 | (s) | Attribute: | Embedded Data type |
| 3 | (m) | Attribute: | Access Privilege |
| 3.1 | (m) | Attribute: | Password |
| 3.2 | (m) | Attribute: | Access Groups |
| 3.3 | (m) | Attribute: | Access Rights |
| 4 | (m) | Attribute: | List of Local Detail |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Information Report |
| 6 | (o) | OpsService: | Information Report List |
| 7 | (o) | OpsService: | Exchange |
| 8 | (o) | OpsService: | Exchange List |

#### 6.2.4.3.3.2 Attributes

**Symbolic Address**

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

**Data type**

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

**Data type ID**

This attribute is the numeric identifier of a STRUCTURE data type.

**Embedded Data type**

This attribute is used to embed the data type description in the record definition. The contents for this attribute are shown in the Data type model.

**Access Privilege**

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

**Password**

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

**Access Groups**

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

**Access Rights**

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

**List of Local Detail**

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

#### 6.2.4.3.3.3 Services

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

### 6.2.4.3.4 Variable list class specification

#### 6.2.4.3.4.1 Formal model

**FAL ASE:** **VARIABLE ASE**
**CLASS:** **VARIABLE LIST**
**CLASS ID:** **10**
**PARENT CLASS:** **TOP**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | (m) | Attribute: | Number of Entries |
| 2 | (m) | Attribute: | List Of Variables |
| 3 | (c) | Attribute: | Deletable |
| 4 | (m) | Attribute: | Access Privilege |
| 4.1 | (m) | Attribute: | Password |
| 4.2 | (m) | Attribute: | Access Groups |
| 4.3 | (m) | Attribute: | Access Rights |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Exchange |

| | | | |
|---|---|---|---|
| 6 | (o) | OpsService: | Exchange List |
| 7 | (o) | OpsService: | Information Report |
| 8 | (o) | OpsService: | Information Report List |

#### 6.2.4.3.4.2 Attributes

**Number of Entries**

This attribute specifies the number of variables in the list.

**List of Variables**

This attribute identifies the variables (only simple, array or record variable objects) by Key Attribute that are contained in the list.

**Deletable**

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

**Access Privilege**

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

**Password**

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

**Access Groups**

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

**Access Rights**

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

Right to Delete for Access Groups

Right to Write for Access Groups

Right to Read for Access Groups

Right to Delete for the registered Password

Right to Write for the registered Password

Right to Read for the registered Password

Right to Delete for all Communication Partners

Right to Write for all Communication Partners

Right to Read for all Communication Partners

#### 6.2.4.3.4.3 Services

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

### 6.2.4.4 Variable ASE service specification

#### 6.2.4.4.1 Supported services

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

Read

Write

Read List

Write List

Information Report

Information Report List

Exchange

Exchange List

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

#### 6.2.4.4.2 Read service

##### 6.2.4.4.2.1 Service overview

This confirmed service may be used to read the value of a variable object or variable list object. It may be used with application relationships configured to support the client/server model, or it may be used with application relationships configured to support the publisher/subscriber "pull" model.

##### 6.2.4.4.2.2 Service primitives

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

**Table 41 – Read service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variable Specifier | M | M (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Data type Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| Best Effort Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Access Results | | | M | M (=) |
| Error Status | | | S | S (=) |
| Returned Data | | | S | S (=) |
| Value | | | M | M (=) |
| Data type | | | C | C (=) |
| Object Revision | | | C | C (=) |
| Timeliness | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

This parameter carries the parameters of the service invocation.

**Variable Specifier**

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

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Component Identifier**

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify a component with a nesting level greater than one if supported by the AP.

**Numeric Address/Data Length**

This parameter provides the numeric address and the data length of the data to be read. The mapping between this parameter and the actual memory address in a real system is a local matter.

**Data type Requested**

This optional parameter indicates that the data type of each variable should be returned with the data.

**Object Revision Requested**

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

**Best Effort Requested**

This conditional, optional parameter is used to request to perform a “best effort” read of a variable list. In the “best effort” read, the service succeeds if at least one variable in the variable list can be read.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**List of Access Result**

This parameter specifies the responses returned by the remote AP. If “best effort” was requested for a variable list, then a list of responses is returned. The order of the response in the list is the same as the order of the variables in the variable list.

**Error Status**

This selection type parameter is when “best effort” was requested for a variable list and one of the variables in the list could not be read. It indicates the reason for the read failure.

**Returned Data**

This selection type parameter specifies the data returned by the remote AP. It is always present if “best effort” was not requested for a variable list.

**Value**

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

**Data type**

This optional parameter indicates that the data type of each value is returned.

**Object Revision**

This conditional parameter specifies the object revision of the specified object. It is present if the object revision was requested.

**Timeliness**

This conditional parameter indicates the data-link layer timeliness status for the referenced variable, or variable list object. This parameter is present if it is supported on the specified AR.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.4.4.2.3 Service procedure

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

The Read Service is an "all or nothing" service. All or nothing means that the service succeeds only if the requested value, or values in the case of a variable list object, are read and returned.

If the object specified to read is a variable list object, the value, data type (when requested), and object revision (when requested) of the variables in the variable list are returned.

A timeliness parameter is included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

### 6.2.4.4.3 Read list service

#### 6.2.4.4.3.1 Service overview

This confirmed service is used to read the values of multiple variables or a single variable list object. It may be used with application relationships configured to support the client/server model.

It operates in a "best effort" fashion, such that the service succeeds if the value for at least one variable or one variable in a variable list is returned, otherwise it fails. When the variable was successfully accessed, the returned data should be the variable value otherwise it should be an error status.

#### 6.2.4.4.3.2 Service primitives

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

**Table 42 – Read list service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Data type Requested | U | U (=) | | |
| List Of Variable Specifiers | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID(s) | S | S (=) | | |
| List Of Numeric Addresses and Data Lengths | S | S (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Access Results | | | M | M (=) |
| List of Data types | | | C | C (=) |
| List of Data | | | M | M (=) |
| Error Status | | | S | S (=) |
| Value | | | S | S (=) |
| Value with Object Revision | | | S | S (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

This parameter carries the parameters of the service invocation.

**Data type Requested**

This optional parameter is used to request that the type description of the List of data parameter be returned. A value of TRUE indicates that the type description is desired. A value of FALSE indicates that type description is not desired. This request can be used only with List of Variable Specifiers selection.

**List of Variable Specifiers**

This parameter individually identifies a list of variables and/or components of variables to be read or one variable list to be read. The nesting level of the components to be read can be equal to or greater than one.

**Key Attribute**

This selection type parameter specifies a Variable or Variable list Identifier key attribute values.

**Key Attribute And Component ID(s)**

This parameter identifies the components of a constructed data type variable by a combination of one of its key attributes and the components path ID(s).

**List Of Numeric Addresses and Data Lengths**

This parameter provides the numeric address and the data length of the variable objects to be read. The mapping between this parameter and the actual memory address in a real system is a local matter. This selection is allowed to access only the data of variable object

NOTE The Numeric Address parameters may be useful to specify a variable or a certain memory location in a called service user. These addresses are not global. One example of the use of the parameters is to specify a pointer to a target memory location.

**Object Revision Requested**

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired for all variables (if supported). A value of FALSE indicates that object revision is not desired for any variable.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**List of Access Results**

This parameter specifies an indicator for each variable accessed. If an access succeeds the indicator should be TRUE otherwise it should be FALSE.

**List of Data types**

This optional parameter provides the type description of the values in the List of Data parameter.

**List of Data**

This parameter specifies one or more error statuses, variable values, or variable values with object revision read. The error statuses or values with/without object revision are concatenated in the order in which they were described in the request. For variable lists this is always the order of the list definition. For each variable or a variable in a variable list successfully accessed, the returned data should be the value or the value with object revision of the variable object (if requested). Otherwise it should be an error status. The returned data should also be an error status if the object revision is requested and the variable object accessed does not support the object revision attribute.

**Error Status**

This parameter indicates the most probable reason why the operation failed using the error class and error code defined for the read service.

**Value**

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

**Value with Object Revision**

This parameter specifies a value read plus the object revision of the variable object. For variables, this parameter specifies the value of the variable. For variable lists, this parameter specifies the values of each of the variables in the list concatenated in the order in which they appear in the list.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.4.4.3.3 Service procedure

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

The Read List Service is a "best effort". Best effort means that the service succeeds if at least one value is read.

If the user is not able to read at least one of the values, the service fails and the user issues a Read List Service response (-) primitive indicating the reason.

When requested, the data type descriptions are returned. When requested, the object revision attribute value is returned concatenated with the value of the variable.

#### 6.2.4.4.4 Write service

##### 6.2.4.4.4.1 Service overview

This confirmed service is used to write the value of a variable. It is not used with the publisher/subscriber model.

##### 6.2.4.4.4.2 Service primitives

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

**Table 43 – Write service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variable Specifier | M | M (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Value | M | M (=) | | |
| Data type | U | U (=) | | |
| Object Revision | U | U (=) | | |
| Best Effort Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Error Status | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Variable Specifier**

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

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Component Identifier**

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify a component with a nesting level greater than one if supported by the AP.

**Numeric Address/Data Length**

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

**Value**

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

NOTE If any variable in a variable list object cannot be updated, none of the variables in the variable list object will be updated, and the write will fail.

**Data type**

This optional parameter specifies the data type of the values in the Value parameter.

**Object Revision**

This optional parameter specifies the expected object revision of the variable object. When present, it indicates that the AP containing the variable is to compare the value of this parameter to the Object Revision of the variable before applying the write.

**Best Effort Requested**

This conditional, optional parameter is used to request to perform a “best effort” write of a variable list. In the “best effort” write, the service succeeds if at least one variable in the variable list can be written.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**List of Status**

This parameter specifies more than one status if a “best effort” write was requested. The status indicates success or the reason for failure. The order in the list is the same as the order of the variables in the variable list.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.4.4.4.3 Service procedure

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

The Write Service is an "all or nothing" service. All or nothing means that the service succeeds only if the requested value, or values in the case of a variable list object, are successfully written.

### 6.2.4.4.5 Write list service

#### 6.2.4.4.5.1 Service overview

This confirmed service is used to write the values of multiple variables or variables in a single variable list object. The number of variables referenced or defined in the list is a device local matter. It may be used with application relationships configured to support the client/server model.

It operates in a "best effort" fashion, such that the service succeeds if the value for at least one variable or variable in a variable list is written, otherwise it fails.

#### 6.2.4.4.5.2 Service primitives

The service parameters for this service are shown in Table 44.

**Table 44 – Write list service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List of Variable Specifiers | S | S (=)) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID(s) | S | S (=) | | |
| List of Numeric Addresses and Data Lengths | S | S (=) | | |
| List of Data types | U | U (=) | | |
| List of Data | M | M (=) | | |
| Value | S | S (=) | | |
| Value with Object Revision | S | S (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Variable Data Access Statuses | | | M | M (=) |
| List of Error Statuses | | | C | C (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

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

**List of Variable Specifiers**

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be written. The nesting level of the components to be written can be equal to or greater than one.

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Component ID(s)**

This parameter identifies components of a constructed data type variable by a combination of one of its key attributes and the component path ID(s).

**List of Numeric Addresses and Data Lengths**

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

**List of Data types**

This optional parameter provides the type description of the values in the List of Data parameter.

**List of Data**

This parameter specifies one or more values or one or more values with object revision to write. The values with/without object revision are concatenated in the order in which they are described in the preceding parameters. For variable lists this is always the order of the list definition.

**Value**

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

**Value with Object Revision**

This parameter specifies a value to write plus the expected object revision of the variable object. For variables, this parameter specifies the value of the variable. For variable lists, this parameter specifies the values of each of the variables in the list concatenated in the order in which they appear in the list.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**List of Variable Data Access Statuses**

This parameter specifies an indicator for each variable accessed. If an access succeeds, the indicator should be TRUE, otherwise it should be FALSE.

**List of Error Statuses**

This parameter specifies an error status for each individual variable or variable in variable list for which the access failed. The order of the error statuses is the same as the order of the objects identified in the service request.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.4.4.5.3 Service procedure

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

The Write List Service is a "best effort" service. Best effort means that the service succeeds if at least one value is written.

If the responding user is able to write the value for at least one of the requested variables, the user returns a Write List Service response (+) primitive.

If the user is not able to write at least one of the values, the user issues a Write List Service response (-) primitive indicating the reason.

### 6.2.4.4.6 Information report service

#### 6.2.4.4.6.1 Service overview

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

#### 6.2.4.4.6.2 Service primitives

The service parameters for this service are shown in Table 45.

**Table 45 – Information report service**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Variable Specifier | C | C (=) |
| Key Attribute | S | S (=) |
| Key Attribute and Component ID | S | S (=) |
| Numeric Address and Data Length | S | S (=) |
| Value | M | M (=) |
| Object Revision | U | U (=) |
| Data type | U | U (=) |
| Timeliness | | C |
| Duplicate FAL PDU Body | | C |

**Argument**

This parameter carries the parameters of the service invocation.

**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 gives the remote address to which the requested Information Report should be sent.

**Source DL-Address**

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

**Variable Specifier**

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

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Component Identifier**

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify a component with a nesting level greater than one if supported by the AP.

**Numeric Address and Data Length**

This parameter provides the numeric address and data length reported.

**Value**

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

**Object Revision**

This optional parameter specifies the expected object revision of the variable object. When present, it indicates that the calling service user wishes the server to check the current Object Revision parameter of the variable to be updated. If supported, the server only updates the value of a variable in the list if its object revision attribute is equivalent to the value of this parameter.

**Data type**

This optional parameter specifies the data type of the values in the Value parameter.

**Timeliness**

This conditional parameter indicates the data-link layer timeliness status for the referenced variable, or variable list object. It is present if timeliness is supported on the specified AR.

**Duplicate FAL PDU Body**

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the data-link layer. It is present if supported in the DL Mapping Attributes of the AREP.

#### 6.2.4.4.6.3 Service procedure

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

A timeliness parameter is included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

### 6.2.4.4.7 Information report list service

#### 6.2.4.4.7.1 Service overview

This unconfirmed service is used by an application process to report a list of values or a list of values with object revision to receiver(s) designated by the AR. The number of variables referenced or defined in the list is a device local matter.

#### 6.2.4.4.7.2 Service primitives

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

**Table 46 – Information report list service**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| List of Variable Specifiers | S | S (=) |
| Key Attribute | S | S (=) |
| Key Attribute and Component IDs | S | S (=) |
| List of Numeric Addresses and Data Lengths | S | S (=) |
| List of Data types | U | U (=) |
| List of Data | M | M (=) |
| Value | S | S (=) |
| Value with Object Revision | S | S (=) |
| Timeliness | | C |
| Duplicate FAL PDU Body | | C |

**Argument**

This parameter carries the parameters of the service invocation.

**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 gives the remote address to which the requested Information Report List 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 Information Report List is to be sent.

**List of Variable Specifiers**

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be written. The nesting level of the components to be written can be equal to or greater than one.

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Component ID(s)**

This parameter identifies components of a constructed data type variable by a combination of one of its key attributes and the components path ID(s).

**List of Numeric Addresses and Data Lengths**

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

**List of Data types**

This optional parameter provides the type description of the values in the List of Data parameter.

**List of Data**

This parameter specifies one or more values or one or more values with object revision to write. The values with/without object revision are concatenated in the order in which they are described in the preceding parameters. For variable lists this is always the order of the list definition.

**Value**

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

**Value with Object Revision**

This parameter specifies a value to write plus the expected object revision of the variable object. For variables, this parameter specifies the value of the variable. For variable lists, this parameter specifies the values of each of the variables in the list concatenated in the order in which they appear in the list.

**Timeliness**

This conditional parameter indicates the data-link layer timeliness for the list of values. It is present if timeliness is defined for the specified AR.

**Duplicate FAL PDU Body**

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the data-link layer. It is present if supported in the DL Mapping Attributes of the AREP.

#### 6.2.4.4.7.3 Service procedure

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

If the object specified to be reported is a variable list object, the values or the values with object revision of the variables in the variable list are concatenated in the List of Data parameter in the order that the variables appear in the variable list.

A timeliness parameter is included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

### 6.2.4.4.8 Exchange service

#### 6.2.4.4.8.1 Service overview

This confirmed service is used to write the value of one variable or variable list object, and read the value of another variable or a variable list object in a single operation. The order of processing the read and write by the responding user is not specified by the FAL.

#### 6.2.4.4.8.2 Service primitives

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

**Table 47 – Exchange service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variable to Write | M | M (=) | | |
| Specifier | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Element ID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Value | M | M (=) | | |
| Data type | U | U (=) | | |
| Object Revision | U | U (=) | | |
| Variable to Read | M | M (=) | | |
| Specifier | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and ElementID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Data type Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Write Result | | | M | M (=) |
| Read Response | | | M | M (=) |
| Error Status | | | S | S (=) |
| Returned Value | | | S | S (=) |
| Value | | | M | M (=) |
| Data type | | | C | C (=) |
| Object Revision | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Specifier for Variable to Write**

This parameter specifies the variable, variable list, or variable field to be updated.

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Element Identifier**

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify an element with a nesting level greater than one if supported by the AP.

#### Numeric Address/Data Length

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

### Value to Write

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

NOTE If any variable in a variable list cannot be updated, none of the variables in the variable list object will be updated, and the write will fail.

### Data type

This optional parameter specifies the data type of the values to write in the Value parameter.

### Object Revision

This optional parameter specifies the expected object revision of the variable object to be updated. When present, it indicates that the AP containing the variable is to compare the value of this parameter to the Object Revision of the variable before applying the write.

### Variable Specifier

This selector type parameter specifies the variable, variable list, or variable field to be read.

#### Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

#### Key Attribute and Element Identifier

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify an element with a nesting level greater than one if supported by the AP.

#### Numeric Address/Data Length

This parameter provides the numeric address and the data length of the data to be read. The mapping between this parameter and the actual memory address in a real system is a local matter.

### Data type Requested

This optional parameter indicates that the data type of each variable should be returned with the data.

### Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

### Result (+)

The Result (+) parameter indicates that the service request succeeded.

### Write Status

This parameter indicates whether or not the variable was updated as requested. If it was not updated, it indicates the most probable reason why the operation failed using the error class and error code defined for the write service.

**Read Response**

This parameter provides the data value corresponding to the Variable Specifier to Read parameter in the request primitive. This parameter indicates the result of the read operation.

**Error Status**

This parameter is used to indicate that the access to the specified object was not successful. It indicates the most probable reason why the operation failed using the error class and error code defined for the read service.

**Returned Value**

This parameter specifies the value of the referenced variable or variable list object. If the identified variable was a variable list object, then this parameter will contain a concatenated set of values for each variable in the list.

**Value**

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

NOTE Because there are no identifiers in a concatenated list of values for a variable list object, a read failure for any variable in the variable list object causes the service to fail.

**Data type**

This optional parameter indicates that the data type of each value returned.

**Object Revision**

This conditional parameter specifies the object revision of the specified object. It is present if the object revision was requested.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.4.4.8.3 Service procedure

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

The Exchange Service is a "best effort" service. Best effort means that the service succeeds if either of the requested read or write operations succeeds.

If a variable list was specified for the variable to write, the value and object revision (when present) of each variable in the variable list are concatenated in the Exchange Request APDU Body in the order that the variables appear in the variable list.

If a variable list was specified for the variable to read, the value and object revision (when present) of each variable in the variable list are concatenated in the Exchange Response APDU Body in the order that the variables appear in the variable list.

### 6.2.4.4.9 Exchange list service

#### 6.2.4.4.9.1 Service overview

This confirmed service is used to write multiple variable objects or single components of multiple variable objects or a single variable list object and to read multiple variable objects or

single components of multiple variable objects or a single variable list object in a single operation. The order of processing the read and write by the responding user is not specified by the FAL.

#### 6.2.4.4.9.2 Service primitives

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

**Table 48 – Exchange list service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variables to Write | M | M (=) | | |
| List of Variable Specifiers | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component IDs | S | S (=) | | |
| List of Numeric Addresses and Data Lengths | S | S (=) | | |
| List of Data types | U | U (=) | | |
| List of Data | M | M (=) | | |
| Value | S | S (=) | | |
| Value with Object Revision | S | S (=) | | |
| Variables to Read | M | M (=) | | |
| List of Variable Specifiers | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component IDs | S | S (=) | | |
| List of Numeric Addresses and Data Lengths | S | S (=) | | |
| Data type Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Write Access Results | | | M | M (=) |
| List of Error Statuses | | | C | C (=) |
| List of Read Access Results | | | M | M (=) |
| List of Data types | | | C | C (=) |
| List of Data | | | M | M (=) |
| Error Status | | | S | S (=) |
| Value | | | S | S (=) |
| Value With Object Revision | | | S | S (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Variables to Write**

This parameter specifies information related to the variables to be updated.

**List of Variable Specifiers**

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be written. The nesting level of the components to be written can be equal to or greater than one.

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Component ID(s)**

This parameter identifies the component of a constructed data type variable by a combination of one of its key attributes and the components path ID(s).

**List of Numeric Addresses and Data Lengths**

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

**List of Data types**

This optional parameter provides the type description of the values in the List of Data parameter.

**List of Data**

This parameter specifies one or more values or one or more values with object revision to write. The values with/without object revision are concatenated in the order in which they are described in the preceding parameters. For variable lists this is always the order of the list definition.

**Value**

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

**Value with Object Revision**

This parameter specifies a value to write plus the expected object revision of the variable object. For variables, this parameter specifies the value of the variable. For variable lists, this parameter specifies the values of each of the variables in the list concatenated in the order in which they appear in the list.

**Variables to Read**

This parameter carries information related to the variables to be read.

**List of Variable Specifiers**

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be read. The nesting level of the components to be read can be equal to or greater than one.

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Key Attribute and Component ID(s)**

This parameter identifies the component of a constructed data type variable by a combination of one of its key attributes and the component path ID(s).

#### List of Numeric Addresses and Data Lengths

This parameter provides the numeric address and the data length of the data to be read. The mapping between this parameter and the actual memory address in a real system is a local matter.

#### Data type Requested

This optional parameter is used to request that the data type be returned. A value of TRUE indicates that data type is desired. A value of FALSE indicates that data type is not desired.

#### Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired.

### Result (+)

The Result (+) parameter indicates that the service request succeeded.

#### List of Write Access Results

This parameter specifies an indicator for each variable accessed. If an access succeeds the indicator should be TRUE otherwise it should be FALSE.

#### List of Error Statuses

This parameter specifies an error status for each individual variable or variable in variable list for which the access failed. The order of the error statuses is the same as the order of the objects identified in the service request.

#### List of Read Access Results

This parameter specifies an indicator for each variable accessed. If an access succeeds the indicator should be TRUE otherwise it should be FALSE.

#### List of Data types

This optional parameter provides the type description of the values in the List of Data parameter.

#### List of Data

This parameter specifies one or more error statuses, values or values with object revision read. The error statuses or values with/without object revision are concatenated in the order in which they were described in the request. For variable lists this is always the order of the list definition. For each variable or a variable in a variable list successfully accessed, the returned data should be the value or the value with object revision of the variable object (if requested). Otherwise it should be an error status. The returned data should also be an error status if the object revision is requested and the variable object accessed does not support the object revision attribute.

#### Error Status

This parameter indicates the most probable reason why the operation failed using the error class and error code defined for the read service.

#### Value

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

#### Value with Object Revision

This parameter specifies a value read plus the object revision of the variable object. For variables, this parameter specifies the value of the variable. For variable lists, this

parameter specifies the values of each of the variables in the list concatenated in the order in which they appear in the list.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.4.4.9.3 Service procedure

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

The Exchange Service is a "best effort" service. Best effort means that the service succeeds if either of the requested read or write operations succeeds.

The write operation succeeds if the access to any variables of the list or any variable in the variable list succeeds.

The read operation succeeds if any variable accessed in the list or variables in the variable list succeeds.

### 6.2.5 Event ASE

#### 6.2.5.1 Overview

The FAL event model defines three event related objects, event objects for the definition of events and their messages, event notifiers for the distribution of event messages, and event lists for the acquisition of event summary information from a group of event objects.

Event objects are used to define messages used to report event occurrences. Event messages contain information that identifies and describes occurrences of events. To simplify the processing of event messages, event objects are defined to generate messages using one of four defined formats.

NOTE 1 For example, event objects may be defined that generate event messages containing a time-tag and/or a message count for event occurrences.

Notifiers are responsible for collecting event messages from event objects, and distributing one or more in a single invocation of the FAL event notification service. The number of event messages that may be submitted in a single service invocation is limited by the maximum APDU size that can be transferred by the AR. Figure 4 illustrates this concept.

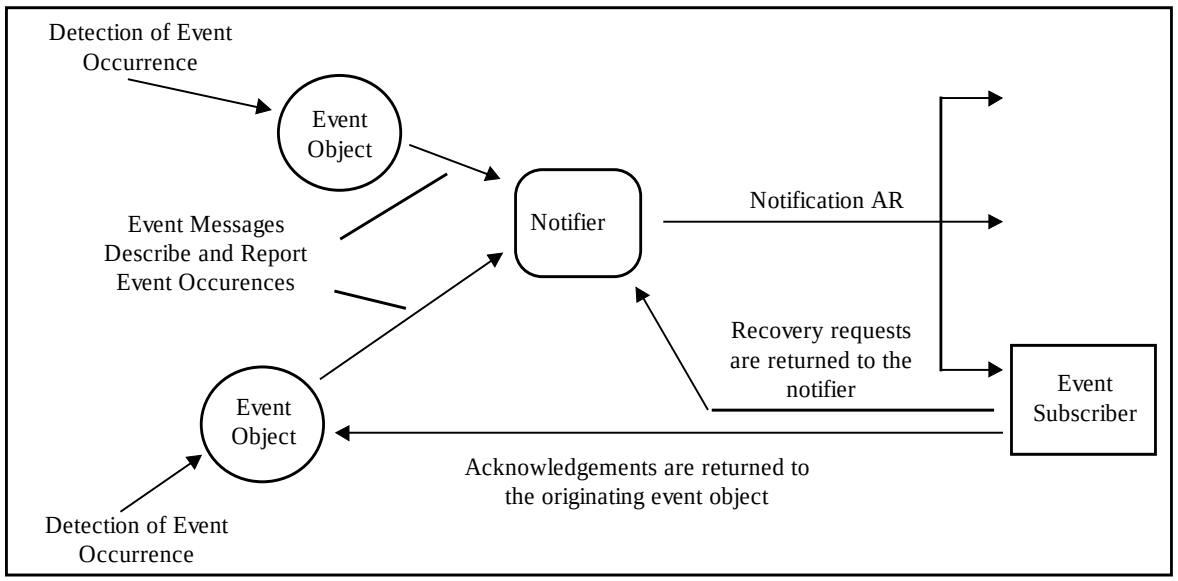



**Figure 4 – Event model overview**

To provide flexibility in the event notification process notifiers may be defined to use one of four types of event notifications to carry event messages. Once defined, a notifier always uses the same type of event notification to carry one or more different types of event messages.

If an application process fails to receive one or more event notifications, a notification recovery service is provided for it to request a retransmission from the notifier.

In addition, the get event summary service is provided to permit an application process to query one or more event objects to select those that meet certain criteria pertaining to the current state of the event objects. An event list object is defined that allows event objects to be grouped together to simplify the query process.

The AR endpoint class used to distribute event notifications is specified by one of the notifier attributes. An event attribute is used to specify which AR distributes the notifications.

Queued ARs are most often used because they prevent event notifications from being overwritten before they are read. When the purpose of the notification is to signal a change in the state of a field device AP, buffered ARs may be used.

NOTE 2 When no events have occurred during a specified period of time notifiers may choose to generate a special "heartbeat" event notification that indicates to subscribers that the notifier is still alive. These special notifications look like the last normal notification sent except that they do not contain the event messages. Heartbeat notifications are defined to permit event subscribers to determine if they have missed an event notification.

In this model, application processes are responsible for providing the functions for event, notifier, and event list objects, and the FAL is responsible for providing communication services designed specifically for them. The application process detects events, builds event messages and aggregates them together. It distributes the aggregated set using the FAL event notification service. At the receiving end, subscriber application processes can use one of two event acknowledgment services to acknowledge event occurrences.

The remainder of this subclause specifies the class definitions and services for event, notifier, and event list objects.

#### 6.2.5.2 Event model class specification

##### 6.2.5.2.1 Event class specification

###### 6.2.5.2.1.1 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **EVENT ASE** |
| **CLASS:** | | EVENT | |
| **CLASS ID:** | | | 4 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Event Status |
| 1.1 | (o) | Attribute: | Active (TRUE,FALSE) |
| 1.2 | (o) | Attribute: | Detected (TRUE,FALSE) |
| 1.3 | (m) | Attribute: | Enabled (TRUE,FALSE) |
| 1.4 | (o) | Attribute: | Acknowledged (TRUE,FALSE) |
| 2 | (m) | Attribute: | Message Type (SIMPLE,<br>REPORTED EVENT COUNT,<br>EVENT DETECTION TIME,<br>COMPOSITE,<br>SIMPLE WITH DATA,<br>REPORTED EVENT COUNT WITH DATA,<br>EVENT DETECTION TIME WITH DATA,<br>COMPOSITE WITH DATA) |
| 3 | (m) | Attribute: | Access Privilege |
| 3.1 | (m) | Attribute: | Password |
| 3.2 | (m) | Attribute: | Access Groups |
| 3.3 | (m) | Attribute: | Access Rights |
| 4 | (o) | Attribute: | Last Reported Event Info |
| 4.1 | (o) | Attribute: | Count |
| 4.2 | (o) | Attribute: | Detection Time |
| 5 | (c) | Constraint: | Message Type = SIMPLE WITH DATA \|<br>REPORTED EVENT COUNT WITH DATA \|<br>EVENT DETECTION TIME WITH DATA \|<br>COMPOSITE WITH DATA) |
| 5.1 | (c) | Attribute: | Event Data Specifier |
| 5.1.1 | (s) | Attribute: | Variable ID |
| 5.1.2 | (s) | Attribute: | Data type ID and Length |
| 6 | (c) | Constraint: | Event Status.Acknowledgment Supported = TRUE |
| 6.1 | (o) | Attribute: | Acknowledgment Data Specifier |
| 6.1.1 | (s) | Attribute: | Variable ID |
| 6.1.2 | (s) | Attribute: | Data type ID and Length |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Acknowledge Event |
| 2 | (o) | OpsService: | Acknowledge Event List |
| 3 | (o) | OpsService: | Enable Event |
| 4 | (o) | OpsService: | Enable Event List |
| 5 | (o) | OpsService: | Get Event Summary |
| 6 | (o) | OpsService: | Get Event Summary List |
| 7 | (o) | OpsService: | Query Event Summary List |

#### 6.2.5.2.1.2 Attributes

**Event Status**

This attribute indicates the status of the event. The status is indicated as a combination of Boolean attributes: active, detected, enabled, acknowledgment support, and acknowledged.

**Active**

This optional attribute indicates, when TRUE, that events are capable of being detected. If FALSE, event occurrences are not detected.

NOTE 1 The notifier object also has an enable attribute used to control whether or not the event notifications are to be generated.

**Detected**

This optional attribute indicates, when TRUE, that the conditions for detecting an event occurrence are TRUE.

**Enabled**

This attribute indicates, when TRUE, that event messages are to be reported to the related notifier upon detection of event occurrences. If FALSE, event messages are not to be generated and reported when event occurrences are detected.

NOTE 2 The notifier object also has an enable attribute used to control whether or not the event notifications are to be generated.

**Acknowledged**

This optional attribute indicates, when TRUE, that all event messages generated for this event object have been acknowledged. This attribute will also be TRUE if no event messages have been generated or if the event object does not support acknowledgment. The value FALSE indicates that the event object is waiting for an acknowledgment.

**Message Type**

This attribute specifies which type of event message is generated when an event occurs. The defined messages types and their structure are:

| Type | Contents |
|---|---|
| SIMPLE MSG | Event Numeric ID |
| REPORTED EVENT COUNT | Event Numeric ID |
| | Last Detected Event Info.Count |
| EVENT DETECTION TIME | Event Numeric ID |
| | Last Reported Event Info.Detection Time |
| COMPOSITE | Event Numeric ID |
| | Last Reported Event Info.Count |
| | Last Reported Event Info.Detection Time |
| SIMPLE MSG WITH DATA | Event Numeric ID |
| | Event Data Identified by the Event Data Specifier |
| REPORTED EVENT COUNT | Event Numeric ID |
| WITH DATA | Last Detected Event Info.Count |
| | Event Data Identified by the Event Data Specifier |
| EVENT DETECTION TIME | Event Numeric ID |
| WITH DATA | Last Reported Event Info.Detection Time |
| | Event Data Identified by the Event Data Specifier |
| COMPOSITE WITH DATA | Event Numeric ID |
| | Last Reported Event Info.Count |
| | Last Reported Event Info.Detection Time |
| | Event Data Identified by the Event Data Specifier |

**Access Privilege**

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

**Password**

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

**Access Groups**

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

**Access Rights**

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

– Right to Acknowledge for registered Password
– Right to Enable/Disable for registered Password
– Right to Acknowledge for the Access Groups
– Right to Enable/Disable for the Access Groups
– Right to Acknowledge for all Communication Partners
– Right to Enable/Disable for all Communication Partners

**Last Reported Event Info**

This optional attribute specifies the count and time of detection of the last reported event.

**Count**

This optional attribute counts the number of reported event occurrences. It is incremented by one for each event occurrence detected and reported. When the event object is disabled, detected event occurrences are not counted.

When the value of this attribute rollsover, it rollsover to 1, skipping 0. The value 0 is always used to indicate that no event messages have been generated. If the message type attribute is set to REPORTED EVENT COUNT or COMPOSITE, its updated value is included in event messages.

NOTE 3 The presence of this attribute is independent of whether it is contained in an event message. Its presence indicates this attribute is accessible using the FAL management services. FAL is not involved with incrementing the value of this attribute. Therefore, the FAL does not know when its value rolls over, or if it rolls over properly.

**Detection Time**

This optional attribute specifies the time that the last reported event was detected. If the message type attribute is set to EVENT DETECTION TIME or COMPOSITE its value is included in event messages.

NOTE 4 The presence of this attribute is independent of whether it is contained in an event message. Its presence indicates this attribute is accessible using the FAL management services.

**Event Data Specifier**

This conditional-type attribute specifies user data to be included in an event message. Event data is specified by either identifying a variable object to be incorporated into an event message, or by simply specifying its data type and length. This attribute is present when the Message Type indicates that event data is to be included in event messages.

**Variable ID**

This selection type attribute identifies a variable object or a variable list object contained in the same AP as the event whose value is to be incorporated into an event message.

**Data type ID and Length**

This selection type attribute identifies the data type and length of the user data that is to be incorporated into an event message. Depending on the data type, the length may be defined by the data type.

**Acknowledgment Data Specifier**

This optional attribute specifies user data to be included in an acknowledgment message. Acknowledgment data is specified by either identifying a variable object to be incorporated into an acknowledgment message, or by simply specifying its data type.

**Variable ID**

This selection type attribute identifies a variable object or a variable list object contained in the same AP as the event whose value is to be incorporated into an acknowledgment message.

**Data type ID and Length**

This selection type attribute identifies the data type and length of the user data that is to be incorporated into an acknowledgment message.

#### 6.2.5.2.1.3 Services

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

**Acknowledge Event**

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

**Acknowledge Event List**

This optional confirmed service is provided to permit a subscriber of an event to acknowledge receipt of a list of event messages generated by one or more specified event objects.

**Enable Event**

This optional service is provided to permit a subscriber of an event to enable an event.

**Enable Event List**

This optional service is provided to permit a subscriber of an event to enable a list of events.

**Get Event Summary**

This optional service provides the ability to retrieve summary information for one event.

**Get Event Summary List**

This optional service provides the ability to retrieve summary information for a list of events.

**Query Event Summary List**

This optional service provides the ability to query a set of events for summary information based on selection criteria.

### 6.2.5.2.2 Event list class specification

#### 6.2.5.2.2.1 Formal model

| | | |
|---|---|---|
| **FAL ASE:** | | **EVENT ASE** |
| **CLASS:** | EVENT LIST | |
| **CLASS ID:** | | 17 |
| **PARENT CLASS:** | | TOP |
| **ATTRIBUTES:** | | |

| | | | |
|---|---|---|---|
| 1 | (m) | Attribute: | Number of Entries |
| 2 | (m) | Attribute: | List Of Events |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Get Event Summary List |
| 2 | (o) | OpsService: | Query Event Summary List |

#### 6.2.5.2.2.2 Attributes

**Number of Entries**

This attribute specifies the number of events in the list.

**List of Events**

This attribute identifies the event objects are contained in the event list.

#### 6.2.5.2.2.3 Services

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

**Get Event Summary List**

This optional service provides the ability to retrieve summary information for a list of events.

**Query Event Summary**

This optional service provides the ability to query a set of events for summary information based on selection criteria.

### 6.2.5.2.3 Notifier class specification

#### 6.2.5.2.3.1 Formal model

This subclause provides formal class definitions for notifier objects. Notifiers provide the ability to send event notifications composed of event occurrence messages from one or more event objects.

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **EVENT ASE** |
| **CLASS:** | NOTIFIER | | |
| **CLASS ID:** | | | 18 |
| **PARENT CLASS:** | | | TOP |

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | (m) | Attribute: | Enabled (TRUE,FALSE) |
| 2 | (m) | Attribute: | Notification AREP Class |
| 3 | (m) | Attribute: | Notification AREP |
| 4 | (m) | Attribute: | Notification Type (BASIC, SEQUENCED, TIME OF NOTIFICATION, COMPOUND) |
| 5 | (o) | Attribute: | Contained Message Type (SIMPLE, REPORTED EVENT COUNT, EVENT DETECTION TIME, COMPOSITE) |
| 6 | (o) | Attribute: | Contained Event Data (TRUE, FALSE) |
| 7 | (o) | Attribute: | Last Notification Sequence Number |
| 8 | (o) | Attribute: | List Of Events |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (m) | OpsService: | Event Notification |
| 2 | (o) | OpsService: | Notification Recovery |

#### 6.2.5.2.3.2 Attribute

**Enabled**

When TRUE, this attribute indicates that the notifier object is enabled. When enabled the notifier object groups event messages from one or more event objects together and submits them in a single invocation of the event notification service to the FAL for conveyance.

When FALSE, the notifier object is disabled. When disabled, the notifier is inactive. It does not issue any event notification service requests.

**Notification AREP Class**

This attribute identifies the class of the AREP required to convey event notifications.

**Notification AREP**

This attribute identifies the AREP configured to convey event notifications. This AREP is also the AREP used for reporting the event notifications generated as the result of an event recovery request.

**Notification Type**

This attribute specifies the type of notification issued by the notifier. The defined notification types are:

**Type Contents**
BASIC Notifier ID
SEQUENCED Notifier ID
Sequence Number
TIME OF NOTIFICATION Notifier ID
Notification Time-Tag
COMPOUND Notifier ID
Sequence Number
Time-Tag

**Contained Message Type**

This attribute specifies the types of event messages that may be contained in notifications issued by the notifier. The event message types are defined by the event object. When an event notification is constructed by this notifier object, the types of event messages that may be contained in the event notification are defined by this attribute. The defined types are:

| Contained Message Type | Description |
|---|---|
| SIMPLE | All event messages in all notifications are of type SIMPLE. |
| REPORTED EVENT COUNT | All event messages in all notifications are of type REPORTED EVENT COUNT. |
| EVENT DETECTION TIME | All event messages in all notifications are of type EVENT DETECTION TIME. |
| COMPOSITE | All event messages in all notifications are of type COMPOSITE. |
| SIMPLE WITH DATA | All event messages in all notifications are of type SIMPLE. |
| REPORTED EVENT COUNT WITH DATA | All event messages in all notifications are of type REPORTED EVENT COUNT WITH DATA. |
| EVENT DETECTION TIME WITH DATA | All event messages in all notifications are of type EVENT DETECTION TIME WITH DATA. |

| | |
|---|---|
| COMPOSITE WITH DATA | All event messages in all notifications are of type COMPOSITE WITH DATA. |

**Contained Event Data**

This attribute, when TRUE, indicates that event messages conveyed by the notifier may contain event data. Event messages for each of the event message types may contain data. This attribute indicates whether or not event messages that are defined for an event object to contain data are capable of being transferred in an event notification generated by this notifier object.

**Last Notification Sequence Number**

The conditional attribute specifies the last sequence number used. It is incremented for each event notification service invocation. When the value of this attribute rolls-over, it rolls-over to 1, skipping 0. The value 0 is always used to indicate that no event notifications have been generated. It is mandatory when the value of notification type is SEQUENCE or COMPOUND.

NOTE The FAL does not check to see if the roll-over has been performed properly.

**List of Events**

This optional attribute identifies the events that are configured for this notifier.

#### 6.2.5.2.3.3 Services

**Event Notification**

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

**Notification Recovery**

This optional service is provided to permit one or more event notification subscribers to request that the notifier resend one or more event notifications that it has retained. If this service is supported, then the Sequence Number attribute is required.

### 6.2.5.3 Event ASE service specification

#### 6.2.5.3.1 Supported services

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

Acknowledge Event

Acknowledge Event List

Enable Event

Enable Event List

Event Notification

Notification Recovery

Get Event Summary

Get Event Summary List

Query Event Summary

#### 6.2.5.3.2 Acknowledge event

##### 6.2.5.3.2.1 Service overview

This service allows the acknowledgment of a reported event.

#### 6.2.5.3.2.2 Service primitives

The service parameters for this service are shown on Table 49.

**Table 49 – Acknowledge event**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Reported Event Count | M | M (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Reported Event Count**

This parameter is used to select a previously generated event message by the reported event count.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.5.3.2.3 Service procedure

The Confirmed Service Procedure specified in 4.6 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 FAL user.

### 6.2.5.3.3 Acknowledge event list service

#### 6.2.5.3.3.1 Service overview

This confirmed service is used by an AP receiving an event message to acknowledge receipt of one or more event messages. The service contains an acknowledgment policy parameter that indicates for each specified event message, whether only that event message is to be acknowledged or whether all event messages that were generated by the same event object

up to and including the specified event are to be acknowledged. The Acknowledge Event List is a best effort service.

#### 6.2.5.3.3.2 Service primitives

The service parameters for this service are shown in Table 50.

**Table 50 – Acknowledge event list service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Acknowledgment Policy | C | C (=) | | |
| Messages to Ack | M | M (=) | | |
| List Of Event ID | M | M (=) | | |
| List Of Message Specifier | U | U (=) | | |
| Reported Event Count | U | U (=) | | |
| Event Detection Time | U | U (=) | | |
| List Of Acknowledgment Data | U | U (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List Of Access Result | | | M | M (=) |
| List Of Error | | | C | C (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Acknowledgment Policy**

This conditional parameter indicates whether only the specified message is to be acknowledged or whether all messages up to and including the specified message are to be acknowledged. It is present only when the List of Messages to Ack parameter identifies a single message.

**Messages to Ack**

This parameter specifies the messages of all events to be acknowledged.

**List of Event ID**

This parameter identifies the events being acknowledged by this service.

**List of Message Specifier**

This optional parameter specifies which messages are to be acknowledged for each event by this service invocation. If this parameter is not present, all messages for the event are to be acknowledged.

If the acknowledgment policy indicates ONLY THE SPECIFIED then only the message identified by this parameter is to be acknowledged.

If the acknowledgment policy indicates UP TO AND INCLUDING, then the message identified by this parameter, and all messages generated before it, are to be acknowledged.

**Reported Event Count**

This optional parameter is used to select a previously generated event message by the reported event count.

**Event Detection Time**

This optional parameter is used to select a previously generated event message by the time of the event detection.

**List of Acknowledgment Data**

This optional parameter specifies the user data to be included in an event acknowledgment in addition to that used to identify the event occurrence. The type of this data is specified by the Acknowledgement Data Specifier attribute of the Event Object.

**Result(+)**

This selection informs the client that the server was able to successfully acknowledge at least one event object among those appearing in the request.

**List of Access Result**

This parameter specifies the failure/success result of all the event objects acknowledged.

**List of Error**

This parameter indicates the reason for failure for each failed acknowledgement.

**Result(–)**

This selection indicates that the service request failed.

#### 6.2.5.3.3.3 Service procedure

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

The association between the events referenced by the Event Acknowledgement service and the corresponding event object in the receiving AP is performed by the FAL user.

### 6.2.5.3.4 Enable event service

#### 6.2.5.3.4.1 Service overview

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

#### 6.2.5.3.4.2 Service primitives

The service parameters for this service are shown in Table 51.

**Table 51 – Enable event**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Enable Flag | M | M (=) | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

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

**Key Attribute**

This parameter identifies the variable or variable list by one of its key attributes.

**Enable Flag**

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.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.5.3.4.3 Service procedure

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

### 6.2.5.3.5 Event notification service

#### 6.2.5.3.5.1 Service overview

This unconfirmed service is used by a notifier of an FAL AP to notify other APs that one or more events have occurred. When this service is used without including a list of event messages, the event notification is regarded as a “heartbeat” event notification. This service is unconfirmed.

#### 6.2.5.3.5.2 Service primitives

The service parameters for this service are shown in Table 52.

**Table 52 – Event notification service parameters**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Notifier ID | C | C (=) |
| Sequence Number | U | U (=) |
| Notification Time | U | U (=) |
| List of Event Messages | U | U (=) |
| List of Event Key Attributes | M | M (=) |
| List of Event Data types | C | C (=) |
| List of EventMessage Contents | C | C (=) |
| Reported Event Count | C | C (=) |
| Event Detection Time | C | C (=) |
| Event Data | C | C (=) |

**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.

**Notifier ID**

This conditional parameter identifies the notifier issuing the event notification. It is present if the AP has more than one notifier defined for it.

**Sequence Number**

This optional parameter is the sequence number for the event notification. It may be used for notification recovery purposes.

**Notification Time**

This optional parameter is the time tag for the event notification.

**List of Event Messages**

This optional parameter specifies the list of event messages that are to be reported. It may contain messages from one or more event objects, each containing the same set of parameters (specified by the Contained Message Type attribute of the Notifier object). The contents of each message are specified by its event object and should be consistent with that specified for the Notifier object. When this parameter is not present, the event notification is treated as a “heartbeat” notification.

**List of Event Key Attributes**

This parameter identifies each of the specific events being acknowledged by this service.

**List of Event Data types**

This conditional, optional parameter indicates the data type of each of the event data parameters. This parameter may be present only if the event data parameter is present. If the event data parameter is present, this parameter may be present, but is not required to be.

**List of Event Message Contents**

This conditional, optional parameter specifies the event messages generated by the event objects.

**Reported Event Count**

This conditional parameter reports the event count for the occurrence being reported. This parameter is present only if it is defined for the message type of the specified event object and if that message type is supported by the specified notifier.

**Event Detection Time**

This optional parameter reports the time of the event detection. This parameter is present only if it is defined for the message type of the specified event object and if that message type is supported by the specified notifier.

**Event Data**

This conditional parameter specifies user data to be included in an event message in addition to that used to identify the event occurrence. This parameter is present only if it is defined for the specified event object and if it is supported by the specified notifier.

#### 6.2.5.3.5.3 Service procedure

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

### 6.2.5.3.6 Enable event list service

#### 6.2.5.3.6.1 Service overview

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

#### 6.2.5.3.6.2 Service primitives

The service parameters for this service are shown in Table 53.

**Table 53 – Enable event list**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List Of Key Attributes | M | M (=) | | |
| List Of Enable Flags | M | M (=) | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List Of Enable Result | | | M | M (=) |
| List Of Error | | | C | C (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

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

**List of Key Attributes**

This parameter identifies each event by one of its key attributes.

**List of Enable Flags**

This parameter, indicates for each event, whether the corresponding event object is to be enabled (when the respective event flag is TRUE), or to be disabled (when the respective event flag is FALSE).

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**List of Enable Result**

This list of Boolean parameter indicates for each event that was to be enabled or disabled, if the Enable or Disable action was successful (Enable Result = TRUE) or failed (Enable Result = FALSE).

**List of Error**

This list of error parameter provides information on the kind of error occurred for each event whose Enable or Disable Action failed.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.5.3.6.3 Service procedure

The Confirmed Service Procedure specified in 4.6 applies to this service. The Enable Eventlist is a “best effort” service.

#### 6.2.5.3.7 Notification recovery service

##### 6.2.5.3.7.1 Service overview

This unconfirmed service is used to request that a specified number of retained event notifications be returned. Notifications are returned using the event notification service.

##### 6.2.5.3.7.2 Service primitives

The service parameters for this service are shown in Table 54.

**Table 54 – Notification recovery service parameters**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Notifier ID | M | M (=) |
| Number To Recover | U | U (=) |
| Sequence Number | 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 is configured with a Remote Address Configuration Type of FREE. It indicates the destination address to which the requested notification recovery request 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 notification recovery request is to be sent.

**Notifier ID**

This parameter identifies the notifier to which this service is directed.

**Number to recover**

This optional parameter specifies the number of event notifications to be re-sent. If this number is greater than the actual number stored, then only the available notifications will be sent. The default for this is one.

**Sequence Number**

This optional parameter specifies the sequence number of the oldest event notification to be re-sent. If not present, the last notification sent is being requested. It may be used only when the specified notifier uses sequence numbers.

##### 6.2.5.3.7.3 Service procedure

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

Recovery is performed by the responding AP by re-issuing Event Notification service requests containing the notification messages to be recovered.

#### 6.2.5.3.8 Get event summary service

##### 6.2.5.3.8.1 Service overview

This confirmed service is used to request summary information for a specified event object.

##### 6.2.5.3.8.2 Service primitives

The service parameters for this service are shown in Table 55.

**Table 55 – Get event summary service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Event ID | M | M (=)) | | |
| Recovered Message Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Event Status | | | M | M (=) |
| Active | | | M | M (=) |
| Detected | | | M | M (=) |
| Enabled | | | M | M (=) |
| Acknowledgment Supported | | | M | M (=) |
| Acknowledged | | | M | M (=) |
| Object Revision | | | C | C (=) |
| Recovered Message | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Event ID**

This mandatory parameter identifies the event object for which the event summary is being requested. An event list object cannot be specified because the members of the list may be dynamically changed by the AP, making it necessary to identify each event in the response.

**Recovered Message Requested**

This optional parameter specifies, when TRUE, that the last message generated by the event is to be returned.

**Object Revision Requested**

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

**Result(+)**

This selection indicates that the service request succeeded.

**Event Status**

This parameter specifies the values of the event status attribute. It indicates the status of the event. The status is indicated as a combination of Boolean parameters: active, detected, enabled, acknowledgment support, and acknowledged.

**Detected**

This Boolean parameter indicates, when TRUE, that the conditions for detecting an event occurrence are TRUE.

**Active**

This Boolean parameter indicates, when TRUE, that events are capable of being detected. If FALSE, event occurrences are not detected.

**Enabled**

This Boolean parameter indicates, when TRUE, that event messages are to be reported to the related notifier upon detection of event occurrences. If FALSE, event messages are not to be generated and reported when event occurrences are detected.

**Acknowledgment Supported**

This Boolean parameter indicates, when TRUE, that this event object supports the acknowledgment of event messages.

**Acknowledged**

This Boolean parameter indicates, when TRUE, that all event messages generated for this event object have been acknowledged. This attribute will also be TRUE if no event messages have been generated or if the event object does not support acknowledgment. The value FALSE indicates that the event object is waiting for an acknowledgment.

**Recovered Message**

This conditional parameter is present if the service request specified message recovery was requested. When present, this parameter specifies the recovered event message.

**Object Revision**

This conditional parameter specifies the object revision of the specified object. It is present if the object revision was requested.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.5.3.8.3 Service procedure

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

The Get Event Summary Service is an "all or nothing" service that operates through a queue or buffer. All or nothing means that the service succeeds only if the requested summary is returned.

### 6.2.5.3.9 Get event summary list service

#### 6.2.5.3.9.1 Service overview

This confirmed service is used to request summary information for a list of specified event object.

#### 6.2.5.3.9.2 Service primitives

The service parameters for this service are shown in Table 56.

**Table 56 – Get event summary list service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Summary Requests | M | M (=)) | | |
| Requested Events | M | M (=)) | | |
| All Events | S | S (=) | | |
| List of Events | S | S (=) | | |
| Recovered Message Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Event Summaries | | | M | M (=) |
| List Of Event ID | | | C | C (=) |
| List Of Access Result | | | M | M (=) |
| List Of Response | | | M | M (=) |
| Error Status | | | S | S (=) |
| Event Summary | | | S | S (=) |
| Event Status | | | M | M (=) |
| Active | | | M | M (=) |
| Detected | | | M | M (=) |
| Enabled | | | M | M (=) |
| Acknowledgment Supported | | | M | M (=) |
| Acknowledged | | | M | M (=) |
| Object Revision | | | C | C (=) |
| Recovered Message | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Summary Requests**

This mandatory parameter specifies a list of events, and the summary information for each, that is being requested. Each entry in the list may identify either an event or an event list object. Each entry in the list also identifies the summary information that is being requested. If an entry identifies an event list, then the summary information requested will apply to all events in the event list.

**Requested Events**

This mandatory parameter indicates that summaries are being requested for all events, or it contains a list that identifies the event objects for which event summaries are being requested.

**All Events**

This selection type parameter indicates that summaries are being requested for all events.

**List of Events**

This selection type parameter is a list that contains the identifiers of the events and/or event lists for which event summaries are being requested. When selected, the list should contain at least one entry.

**Recovered Message Requested**

This optional parameter specifies, when TRUE, that the last message generated by the event is to be returned.

**Object Revision Requested**

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

**Result(+)**

This selection indicates that the service request succeeded.

**Event Summaries**

This parameter specifies a response for each requested event. The order of the information is the same as the order in which the objects were referenced in the service request.

**List of Event ID**

This conditional parameter identifies each event for which an event summary or error status is returned.

**List of Access Result**

This list of Boolean parameters provides for each event for which an event summary was requested an information, whether the access was successful (Access Result = TRUE) or not (Access Result = FALSE).

**List of Response**

This parameter specifies an error status indicator or the event summary for each requested event. The order of the information is the same as the order in which the objects were referenced in the service request.

**Error Status**

This selection type parameter specifies if the condition which triggers the reporting of the event is active. A value of TRUE indicates that the underlying condition which triggers the event is active. A value of FALSE indicates that the underlying condition which triggers the event is not active.

**Event Summary**

This selection type parameter specifies the summary information for the specified event.

**Event Status**

This parameter specifies the values of the event status attribute. It indicates the status of the event. The status is indicated as a combination of Boolean parameters: active, detected, enabled, acknowledgment support, and acknowledged.

**Detected**

This Boolean parameter indicates, when TRUE, that the conditions for detecting an event occurrence are TRUE.

**Active**

This Boolean parameter indicates, when TRUE, that events are capable of being detected. If FALSE, event occurrences are not detected.

**Enabled**

This Boolean parameter indicates, when TRUE, that event messages are to be reported to the related notifier upon detection of event occurrences. If FALSE, event messages are not to be generated and reported when events occurrences are detected.

**Acknowledgment Supported**

This Boolean parameter indicates, when TRUE, that this event object supports the acknowledgment of event messages.

**Acknowledged**

This Boolean parameter indicates, when TRUE, that all event messages generated for this event object have been acknowledged. This attribute will also be TRUE if no event messages have been generated or if the event object does not support acknowledgment. The value FALSE indicates that the event object is waiting for an acknowledgment.

**Recovered Message**

This conditional parameter is present if the service request specified message recovery was requested. When present, this parameter specifies the recovered event message.

**Object Revision**

This conditional parameter specifies the object revision of the specified object. It is present if the object revision was requested.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.5.3.9.3 Service procedure

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

The Get Event Summary List Service is a "best effort" service that operates through a queue or buffer. Best effort means that the service succeeds if at least one event summary is returned. If the user is not able to read at least one of the values, the service fails and the user issues a Get Event Summary List Service response (-) primitive indicating the reason.

#### 6.2.5.3.10 Query event summary list service

##### 6.2.5.3.10.1 Service overview

This confirmed service provides the means for an application to request that all events, an event list, or a list of events be searched according to the criterion provided. The list of valid criteria supported by the responding application process is defined by the List of Event Summary Selection Criteria attribute of the responding AP. A supported criterion may define either an individual event characteristic, or a combination of event characteristics.

NOTE The selection criteria attribute defined for an event may be based on an event characteristic such as its status, whether it is enabled or disabled, or whether it is has been acknowledged. It is also possible to specify user-defined criteria such as the events related to a particular operating unit or a piece of equipment.

##### 6.2.5.3.10.2 Service primitives

The service parameters for this service are shown in Table 57.

**Table 57 – Query event summary list service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Events to Search | M | M (=)) | | |
| All Events | S | S (=) | | |
| List of Events | S | S (=) | | |
| Selection Criterion | M | M (=) | | |
| Recovered Message Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Event Summaries | | | M | M (=) |
| List Of Event ID | | | C | C (=) |
| List Of Access Result | | | M | M (=) |
| List Of Response | | | M | M (=) |
| Event Status | | | M | M (=) |
| Active | | | M | M (=) |
| Detected | | | M | M (=) |
| Enabled | | | M | M (=) |
| Acknowledgment Supported | | | M | M (=) |
| Acknowledged | | | M | M (=) |
| Object Revision | | | C | C (=) |
| Recovered Message | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Events to Search**

This parameter identifies the event objects that are to be searched. The events are identified by specifying all events, an event list, or a list of events.

**All Events**

This selection type parameter indicates that all events are to be searched.

**List of Events**

This selection type parameter specifies events that are to be searched. The events may be specified as a list of event objects, a list of event list objects, or a combination of the two. When selected, the list should contain at least one entry.

**Selection Criterion**

This parameter identifies the criterion to use in the selection process. The value used should be contained in the List of Event Summary Selection Criteria attribute of the AP containing the events to be searched.

**Recovered Message Requested**

This optional parameter specifies, when TRUE, that the last message generated by the event is to be returned.

**Object Revision Requested**

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

**Result(+)**

This selection indicates that the service request succeeded.

**Event Summaries**

This parameter specifies the summary information for each selected event

**List of Event ID**

This conditional attribute is present when the selection criterion is not “none” or when the selection criterion is “none” and the specified events are “all” or a “list of events”.

**List of Access Result**

This list of Boolean parameters provides for each event for which an event summary is returned an information, whether the access was successful (Access Result = TRUE) or not (Access Result = FALSE).

**List of Response**

This parameter specifies an error status indicator or the event summary for each returned event summary. The order of the information is the same as the order in which the objects were referenced in the service request.

**Event status**

This parameter specifies, when TRUE, that the conditions for detecting an event are currently TRUE. This parameter has the value FALSE if the conditions are FALSE or if the event object is not active.

**Active**

This parameter specifies, when TRUE, that event detection is active for the event. When active, the event object detects when events occur. When inactive, it does not.

**Enabled**

This parameter specifies the value of the Enabled attribute of the identified Event object.

**Unacknowledged**

This parameter, when TRUE, specifies that the event object does not have outstanding event occurrences for which it is awaiting acknowledgments, or that its does not support acknowledgment. If the Event is waiting for acknowledgments this parameter has a value of FALSE.

**Recovered Message**

This conditional parameter is present if the service request specified message recovery was requested. When present, this parameter specifies the recovered event message.

**Object Revision**

This conditional parameter is present if the service request specified the object revision was requested. When present, this parameter specifies the object revision of the event object selected.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.5.3.10.3 Service procedure

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

The Query Event Summary Service is a "best effort" service that operates through a queue or buffer. Best effort means that the service succeeds if at least one event summary is returned. If the user is not able to obtain the summary for at least one event, the service fails and the user issues a Query Event Summary Service response (-) primitive indicating the reason.

### 6.2.6 Load region ASE

#### 6.2.6.1 Overview

A Load Region 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.

Load regions may represent an unnamed volatile memory area, such as that implemented by dynamic computer memory, or a named non-volatile memory object, such as a file. The contents of a load region are referred to as a load image. Load images may contain programs or data. The transfer of a load image to or from a load region is performed using the load process.

To maintain integrity, only one load process for a load region is permitted at a time. The Load Region State attribute of the load region is used to indicate whether the load region is empty, being downloaded, or loaded (see the definition of the attribute for a complete list of the states). Load regions may be cleared using the Discard service.

The Upload State attribute indicates the progress of an upload. This attribute is defined to separate the state of the contents of the load region from the operation of having the contents uploaded (dumped).

This load region model provides services that permit an AP to initiate the download or upload of one of its load regions. It identifies the load region and whether to upload or download as parameters of the request. A third parameter indicates whether the transfer of load image segments is to be accomplished using the Pull Segment service or the Push Segment service.

NOTE The Push and Pull Segment services have no relationship to the Push and Pull AREP types.

Load image segments are pulled by having the receiver of the load image issue requests for them. The AP that contains the load image responds by returning the requested segment, indicating with a parameter when the final segment of the load image has been returned.

Load image segments are pushed by having the AP containing the load image send individual segments to the receiver and wait for a response that indicates whether the segment was received. In this case, the sender indicates when the last segment has been transferred.

After the last segment has been received, the AP that initiated the transfer ends the load process by issuing a terminate request to the remote AP. The terminate request may also be used by either AP when an AP determines that the load process cannot be completed successfully.

#### 6.2.6.2 Load region model specification

##### 6.2.6.2.1 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **LOAD REGION ASE** |
| **CLASS:** | LOAD REGION | | |
| **CLASS ID:** | | | 2 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Load Region Size |
| 2 | (m) | Attribute: | Access Privilege |
| 2.1 | (m) | Attribute: | Password |
| 2.2 | (m) | Attribute: | Access Groups |
| 2.3 | (m) | Attribute: | Access Rights |
| 3 | (m) | Attribute: | Local Address |
| 4 | (m) | Attribute: | Load Region State |
| 5 | (m) | Attribute: | Upload State |
| 6 | (m) | Attribute: | Number of Related Objects In Use |
| 7 | (o) | Attribute: | List of Related Objects |
| 7.1 | (m) | Attribute: | ClassID |
| 7.2 | (m) | Attribute: | Numeric ID |
| 7.2 | (m) | Attribute: | In Use Flag (TRUE, FALSE) |
| 8 | (o) | Attribute: | Contents Size |
| 9 | (o) | Attribute: | Load Image Name |
| 10 | (o) | Attribute: | Fixed Length Segments ( TRUE/FALSE) Default: FALSE |
| 11 | (c) | Constraint: | Fixed Length Segments = TRUE |
| 11.1 | (o) | Attribute: | Fixed Segment Length |
| 12 | (o) | Attribute: | Intersegment Request Timeout |
| 13 | (o) | Attribute: | Sharable |
| 14 | (o) | Attribute: | Local Detail |
| **SERVICES:** | | | |
| 1 | (m) | Ops Service: | Initiate Load |
| 2 | (o) | Ops Service: | Push Segment |
| 3 | (o) | Ops Service: | Pull Segment |
| 4 | (m) | Ops Service: | Terminate Load |
| 5 | (o) | Ops Service: | Discard |

#### 6.2.6.2.2 Attributes

**Load Region Size**

This attribute specifies the maximum size of the Load Region in octets.

**Access Privilege**

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

**Password**

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

**Access Groups**

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

**Access Rights**

This attribute defines the type of access defined for the load region. Valid values are:

- Right to Use in a FI for the Access Groups
- Right to Write for the Access Groups
- Right to Read for the Access Groups
- Right to Use in a FI for the registered Password
- Right to Write for the registered Password
- Right to Read for the registered Password
- Right to Use in a FI for all Communication Partners
- Right to Write for all Communication Partners
- Right to Read for all Communication Partners

**Local Address**

This attribute is a locally significant address of the load region. The value FFFFFFFF hex indicates that no local address is available.

**Load Region State**

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

NOT DOWNLOADABLEThis state indicates that the Load Region is not capable of being downloaded.

DOWNLOADABLE This state indicates that the Load Region is empty, but is capable of being downloaded.

DOWNLOADING This state indicates that the download sequence has been initiated.

DOWNLOAD FAILURE This transient state indicates that the download sequence has failed, but has not yet been terminated.

DOWNLOAD SUCCESS This transient state indicates that the download sequence has succeeded, but has not yet been terminated.

LOADED This state indicates that the download sequence has terminated successfully.

IN-USE This state indicates that the Load Region is loaded and is currently being used.

Download State attribute constraints:

A Load Region object whose contents are stored in non-erasable memory cannot support downloading transfers. Its valid state values are LOADED and IN USE

A Load Region object whose contents are stored in erasable memory supports downloading transfers. Its valid state values are DOWNLOADABLE, DOWNLOADED, DOWNLOADING, DOWNLOADING_FAILURE, DOWNLOADING_SUCCESS, LOADED and IN_USE,

NOTE The state NOT DOWNLOADABLE state is reserved for Load Regions whose contents are stored in erasable memory, but who have encountered a temporary condition that does not allow them to be downloaded. contents of Load Region objects located in an Application Process paying client role.

**Upload State**

This attribute specifies the state of the Upload Process of the Load Region.

NOT UPLOADABLE This state indicates that the Load Region is not capable of being uploaded.

UPLOADABLE This state indicates that the Load Region is capable of being uploaded.

UPLOADING This state indicates that the upload sequence has been initiated.

COMPLETING UPLOAD This transient state indicates that the upload sequence has completed, but has not yet been terminated.

Upload State attribute constraints:

A Load Region object whose contents are stored in erasable memory or not support Uploading transfers. Its valid state values are UPLOADABLE, UPLOADING, and COMPLETING UPLOADING

NOTE The NOT UPLOADABLE state is reserved Load Regions whose contents are stored in erasable memory, but who have encountered a temporary condition that does not allow them to be uploaded.

**Number of Related Objects in Use**

This attribute indicates the number of related Action objects or Function Invocation objects which currently use the contents of the Load Region object.

This attribute is incremented each time a related Function Invocation object enters the RUNNING state or a related Action object is invoked.

This attribute is decremented each time a related Function Invocation object leaves the RUNNING state or a related Action object completes execution.

If the attribute sharable = TRUE, then several Function Invocation objects and Action objects may share the Load Region contents; otherwise only one Function Invocation or Action object can be attached and use the contents of the object

**List of Related Objects**

This optional attribute specifies the Function Invocation and Action objects sharing the contents of the Load Region. Each Function Invocation object or Action object is identified by specifying its class identifier and its numeric identifier.

This list is respectively increased/decreased each time a related Function Invocation object or Action object is created or deleted.

**Class ID**

This attribute is the class identifier for the related object.

**Numeric ID**

This attribute is the numeric identifier for the related object.

**In Use Flag**

This attribute is TRUE if the related object is a Function Invocation object in the RUNNING state or if the related object is an Action object that is currently executing.

**Contents Size**

This optional attribute specifies the length of the data contained in the Load Region. The length is indicated in octets. If the state of the load region is DOWNLOADABLE, then the value of this attribute is zero.

**Load Image Name**

This optional attribute specifies the name of load image contained in the load region.

**Fixed-Length Segments**

This optional attribute specifies, when TRUE, that the load region supports the upload and download of fixed length segments only. The default value for this attribute is FALSE.

**Fixed-Segment Length**

This conditional attribute specifies the length in octets of the fixed-length-segments uploaded to and downloaded from this load region. The attribute is present when the Fixed Length Segments attribute is TRUE.

**Intersegment Request Timeout**

This optional attribute specifies the timeout period for receiving segment requests, in seconds.

**Sharable**

This attribute indicates, when TRUE, that the Load Region object contents may be used by multiple Function Invocation/Action objects. The default value for this attribute is TRUE.

**Local Detail**

This attribute specifies additional information, such as a list of capabilities and/or the number of octets to be transferred.

#### 6.2.6.2.3 Services

**Initiate Load**

This service is used by an application to initiate the download or upload of a Load Region. A parameter of the service indicates whether the load image will be transferred into the Load Region using the Pull Segment or from the Load Region using the Push Segment service.

**Push Segment**

This optional service is used to transfer load segments in the request/indication primitives. Although this service is optional, at least one of the segment transfer services, Push Segment or Pull Segment, is required.

**Pull Segment**

This optional service is used to request that a load segment be returned in the response/confirmation primitives. Although this service is optional, at least one of the segment transfer services, Push Segment or Pull Segment, is required.

**Terminate Load**

This service is used by an AP to terminate the load process.

**Discard**

This optional service is used to clear the contents of the load region.

#### 6.2.6.3 Load region service specification

##### 6.2.6.3.1 Supported services

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

Initiate Load

Terminate Load

Push Segment

Pull Segment

Discard

##### 6.2.6.3.2 Initiate load service

###### 6.2.6.3.2.1 Service overview

This confirmed service is used to request the download or upload of a load region. A parameter of the service indicates whether the load image is to be transferred by the requester using the Pull Segment or Push Segment service. When a download is being requested for a load region in a remote AP (identified by the called load region key attribute) that does not yet exist, the responding AP may either dynamically create the load region and return a positive response, or return a negative response that indicates that the load region does not exist.

###### 6.2.6.3.2.2 Service primitives

The service parameters for this service are shown in Table 58.

**Table 58 – Initiate load service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Region Key Attribute | M | M (=) | | |
| Calling | S | S (=) | | |
| Called | S | S (=) | | |
| Load Type | M | M (=) | | |
| Load Service To Use | M | M (=) | | |
| Load Service Initiator | M | M (=) | | |
| Additional Information | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Additional Detail | | | U | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Called Size | | | C | C (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Load Region Key Attribute**

This parameter identifies the load region to be downloaded or to be uploaded.

**Calling**

This selection type parameter identifies the load region of the requester. It is selected if the requester is the server.

**Called**

This selection type parameter identifies the load region of the responder. It is selected if the requester is the client.

**Load Type**

This parameter specifies whether the load is an UPLOAD (out of the load region) or a DOWNLOAD (into the load region).

**Load Service to Use**

This parameter specifies that the load region is to be transferred from the remote AP using either the Pull Segment or the Push Segment service.

**Load Service Initiator**

This parameter specifies, when TRUE, that the load service identified by the previous parameter is to be initiated by the AP initiating this service. When FALSE, the remote AP is to initiate the load service.

**Additional Information**

This optional parameter specifies additional information, such as a file name and/or the number of octets to be transferred. It may be present if the requester is the server.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Additional Detail**

This optional parameter specifies additional information, such as a list of capabilities and/or the number of octets to be transferred. It may be present if the responder is the server.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Called Size**

This conditional parameter specifies the maximum size in octets that the responding AP is prepared to transfer. It is present if the error was due to a size problem.

#### 6.2.6.3.2.3 Service procedure

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

### 6.2.6.3.3 Terminate load service

#### 6.2.6.3.3.1 Service overview

This confirmed service is used to terminate the load process. It may be used upon successful completion of the load, or to abort a load in progress. It may be requested by either endpoint.

If the load process has completed, it is always issued by the AP that issued the initiate service that started the load process. If the load process is being aborted,

a) the load region sending the image returns to the UPLOADABLE Upload State;

b) the load region receiving the image deletes the incomplete load from its load region and returns to the DOWNLOADABLE State.

#### 6.2.6.3.3.2 Service primitives

The service parameters for this service are shown in Table 59.

**Table 59 – Terminate load service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Region Key Attribute | M | M (=) | | |
| Load Type | M | M (=) | | |
| Load Service Used | M | M (=) | | |
| Terminate Reason | C | C (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Terminate Reason | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Load Region Key Attribute**

This parameter specifies one of the key attributes of the Load Region for which the load is being terminated.

**Load Type**

This parameter indicates the type of load being terminated. Valid values are UPLOAD and DOWNLOAD.

**Load Service Used**

This parameter indicates the type of load service used to transfer load segments. Valid values are PUSH and PULL.

**Terminate Reason**

This conditional parameter indicates the success or failure of the load process. It is present if the server issues the terminate request.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Terminate Reason**

This conditional parameter indicates the success or failure of the load process. It is present if the client issues the terminate request to terminate a download.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.6.3.3.3 Service procedure

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

### 6.2.6.3.4 Push segment service

#### 6.2.6.3.4.1 Service overview

This confirmed service is used after the load process has been initiated to transfer a single load image segment in its request and indication primitives. The response and confirmation primitives are used to convey the success or failure of the segment transfer. This service may be used to support uploads and downloads.

#### 6.2.6.3.4.2 Service primitives

The service parameters for this service are shown in Table 60.

**Table 60 – Push segment service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Type | M | M (=) | | |
| Load Region Key Attribute | M | M (=) | | |
| Segment Number | U | U (=) | | |
| Load Data | M | M (=) | | |
| More Follows | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Load Type**

This parameter specifies whether the load is an UPLOAD (out of the load region) or a DOWNLOAD (into the load region).

**Load Region Key Attributes**

This parameter specifies one of the key attributes of the Load Region whose image is to be transferred.

NOTE The load region is local if this service is being used for an upload. The load region is remote if this service is being used for a download.

**Segment Number**

This optional parameter identifies the number of the segment which is being transferred. Segment numbers are ascending integers beginning with the value one (1).

**Load Data**

This parameter specifies the data to be loaded.

**More Follows**

This parameter indicates 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.

#### 6.2.6.3.4.3 Service procedure

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

### 6.2.6.3.5 Pull segment service

#### 6.2.6.3.5.1 Service overview

This confirmed service is used after the load process has been initiated to request that a specified load image segment be returned in the response and confirmation primitives. This service may be used to support uploads and downloads.

#### 6.2.6.3.5.2 Service primitives

The service parameters for this service are shown in Table 61.

**Table 61 – Pull segment service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Type | M | M (=) | | |
| Load Region Key Attribute | M | M (=) | | |
| Segment Number | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Load Data | | | M | M (=) |
| More Follows | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Load Type**

This parameter specifies whether the load is an UPLOAD (out of the load region) or a DOWNLOAD (into the load region).

**Load Region Key Attribute**

This parameter specifies one of the key attributes of the Load Region whose image is to be transferred.

NOTE The load region is remote if this service is being used for an upload. The load region is local if this service is being used for a download.

**Segment Number**

This optional parameter identifies the number of the segment which is being requested. Segment numbers are ascending integers beginning with the value one (1).

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Load Data**

This parameter specifies the data to be downloaded or uploaded.

**More Follows**

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

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.6.3.5.3 Service procedure

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

### 6.2.6.3.6 Discard service

#### 6.2.6.3.6.1 Service overview

This confirmed service is used to request that the contents of a load region be discarded.

#### 6.2.6.3.6.2 Service primitives

The service parameters for this service are shown in Table 62.

**Table 62 – Discard service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Region Key Attribute | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

The argument contains the parameters of the service request.

**Load Region Key Attribute**

This parameter specifies one of the key attributes of the Load Region object whose contents are to be discarded.

**Result(+)**

This selection indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

#### 6.2.6.3.6.3 Service procedure

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

### 6.2.6.4 Load region state machines

#### 6.2.6.4.1 General

This standard defines state machines for three types of transfers.

Pull_Uploading: The load region contents are retrieved from a Load Region object, segment by segment, by a Client AP.

Pull_Downloading: A load image contained in a Client AP is copied into a Load Region object, segment by segment, at the request of the server AP that contains the load region. That is, the Server AP requests the client to send the segments of the load image. This kind of transfer is allowed for erasable load regions. Its state diagram and transitions table give additional information related to a non-erasable load region contents.

Push_Downloading: A load image contained in a client AP are sent to the Load Region object, segment by segment, by the client AP. Upon receipt of each segment, the Server AP that contains the load region copies it into the load region. This kind of transfer is allowed for erasable load regions. Its state diagram and transitions table give additional information related to a non-erasable load region contents.

Other kinds of transfer are beyond the scope of this standard.

#### 6.2.6.4.2 Pull upload state machine

##### 6.2.6.4.2.1 Description

This kind of transfer uses the following services: Initiate Load, Pull Segment and Terminate Load

A Server may initiate an uploading transfer. In this case it issues an Initiate_Load.Req service primitive to a Client. This primitive tells the Client to initiate uploading transfer for a specified Load Region. Once the client has pulled the contents, it issues an Initiate_Load.Rsp(+/-) to the Server.

A Client may initiate an uploading transfer. In this case it issues an Initiate_Load.Req primitive to a Server. This primitive tells the Server that the contents of a specified load region object will be pulled. Once the client has pulled the contents, it ends the transfer by issuing a Terminate_Load.Req to the Server.

##### 6.2.6.4.2.2 Sequencing of service primitives

Table 63 summarizes the sequencing of service primitives. In the table, T1 and T2 show different options for initiating the upload, one by the server AP (T1) and one by the client AP (T2).

**Table 63 – Pull upload sequencing of service primitives**

| **FAL User(Client)** | | **FAL User(Server)**<br>**Owner of called Load Region** |
|---|---|---|
| Initiate_Load.Ind | ←----(T 1)------ | Initiate_Load.Req |
| Initiate_Load.Req | -------(T2)-----→ | Initiate_Load.Ind (Transfer started) |
| Initiate_Load.Cnf | ←----(T2)------- | Initiate_Load.Rsp |
| Pull_Segment.Req | -------(T3)-----→ | Pull_Segment.Ind (Segments Transferred) |
| Pull_Segment.Cnf | ←----(T3)------- | Pull_Segment.Rsp |
| | | |
| Terminate_Load.Req | -------(T4)-----→ | Terminate_Load.Ind (Transfer ended) |
| Terminate_Load.Cnf | ←----(T4)------- | Terminate_Load.Rsp |
| Initiate_Load.Rsp | -------(T1)-----→ | Initiate_Load.Cnf |

##### 6.2.6.4.2.3 Service parameter value constraints

Table 64 summarizes the service parameter constraints.

**Table 64 – Pull upload service parameter constraints**

| Transition Number | Parameter constraints |
|---|---|
| T1 | Initiate_Load<br>Calling Load Region Key attribute selected,<br>Load Type : = upload,<br>Service To Use : = Pull_Segment,<br>Load Service Initiator : = FALSE |
| T2 | Initiate_Load<br>Called Load Region Key Attribute selected,<br>Load Type : = upload<br>Service To Use : = Pull_Segment<br>Load Service Initiator : = TRUE |
| T3 | Pull_Segment<br>Load Type : = upload,<br>Segment Number : = ordered 1 to n if any<br>More Follows : = TRUE/FALSE with FALSE for the last segment |
| T4 | Terminate_Load<br>Load Type : = upload,<br>Load Service Used : = Pull_Segment |

#### 6.2.6.4.2.4 Upload state transition table

Table 65 gives the upload state transitions.

**Table 65 – Pull upload state table**

| Transition | Current State | Event/Action | Next State |
|---|---|---|---|
| 1 | UPLOADABLE | Initiate_Load.Ind<br>&&Load Region State of this object is in state DOWNLOADABLE II LOADED II IN USE<br><br>Initiate_Load.Rsp(+)<br>the load region image is segmented in accordance with fixed-length attribute constraints into n segments<br>Segments Left To Transfer : = n -- (local counter) | UPLOADING |
| 2 | UPLOADING | Pull_Segment.Ind<br>&& Segments Left To Transfer > 1<br>&&service succeeds(e.g. compliance with Intersegment Req Time Out attribute constraint if supported)<br><br>Pull_Segment.Rsp(+)<br>Segment Number : = Segments Left To Transfer (if any)<br>with More Follows : = TRUE<br>Segments Left To Transfer : = -1 | UPLOADING |
| 3<br>Exist if supports: Intersegmnt Req Time Out | UPLOADING | Pull_Segment.Ind<br>&&Segments Left To Transfer = any value<br>&&service fails(e.g. no compliance with Intersegment Req Time Out attribute constraint if supported)<br><br>Pull_Segment.Rsp(-) | UPLOADING |
| 4 | UPLOADING | Pull_Segment.Ind<br>&&Segments Left To Transfer=1 (last segment)<br>&&service succeeds(e.g. compliance with Intersegment Req Time Out attribute constraint)<br><br>Pull_Segment.Rsp(+)<br>Segment Number : = Segments Left To Transfer (if any)<br>More Follows : = FALSE<br>Segments Left To Transfer : = -1 | COMPLETING UPLOAD |
| 5 | UPLOADING | Terminate_Upload.Ind<br><br>Terminate_Upload.Rsp(-)<br>Error Info : = object state conflict | UPLOADABLE |
| 6 | UPLOADING or COMPLETING UPLOAD | Abort.Ind | UPLOADABLE |
| 7 | COMPLETING UPLOAD | Terminate_Upload.Ind<br><br>Terminate_Upload.Rsp(+) | UPLOADABLE |
| 8 | UPLOADABLE | Initiate_Load.Ind<br>&& Load Region State attribute is not in state: DOWNLOADABLE II LOADED II IN USE<br><br>Initiate_Load.Rsp(-)<br>with Error Info : = object constraint conflict | UPLOADABLE |

#### 6.2.6.4.3 Pull download state machine

##### 6.2.6.4.3.1 Description

This kind of transfer uses the following services: Initiate Load, Pull Segment and Terminate Load.

A Server may initiate a downloading transfer. In this case it issues an Initiate_Load.Req service primitive to a Client. This primitive tells the Client to initiate downloading transfer for a specified Load Region object. Once the Server has pulled the contents, the Client issues an Initiate_Load.Rsp(+/-) to the Server.

A Client may initiate the downloading transfers. In this case, it issues an Initiate_Load.req primitive to a Server. This primitive tells the Server to pull the contents of a specified load region object. Once the Server has pulled the contents, it ends the transfer by issuing a Terminate_Load.Req to the Client

#### 6.2.6.4.3.2 Sequencing of service primitives

Table 66 summarizes the sequencing of service primitives.

**Table 66 – Pull download sequencing of service primitives**

| FAL User(a Client) Owner of a calling Object | | FAL User(a Server) Owner of called Object | |
|---|---|---|---|
| Initiate_Load.Ind | ←----(T 1)------ | Initiate_Load.Req | |
| Initiate_Load.Req | -------(T2)-----→ | Initiate_Load.Ind | (Transfer started) |
| Initiate_Load.Cnf | ←----(T2)------- | Initiate_Load.Rsp | |
| Pull_Segment.Ind | <-------(T3)----- | Pull_Segment.Req | (Transfer continued) |
| Pull_Segment.Rsp | -----(T3)-----→ | Pull_Segment.Cnf | |
| | | | |
| Terminate_Load.Ind | <-------(T4)----- | Terminate_Load.Req | (Transfer ended) |
| Terminate_Load.Rsp | ----(T4)-----→ | Terminate_Load.Cnf | |
| Initiate_Load.Rsp | -------(T1)-----→ | Initiate_Load.Cnf | |

#### 6.2.6.4.3.3 Service parameter constraints

Table 67 summarizes the service parameter constraints.

**Table 67 – Pull download service parameter constraints**

| Transition Number | Parameter constraints |
|---|---|
| T1 | Initiate_Load:<br>Calling Load Region Key attribute selected,<br>Load Type : = download,<br>Service To Use : = Pull_Segment,<br>Load Service Initiator : = TRUE |
| T2 | Initiate_Load<br>Called Load Region Key Attribute selected,<br>Load Type : = download<br>Service To Use : = Pull_Segment<br>Load Service Initiator : = FALSE |
| T3 | Pull_Segment<br>Load Type : = download,<br>Segment Number : = ordered 1 to n if any<br>More Follows : = TRUE/FALSE with FALSE for the last segment |
| T4 | Terminate_Load<br>Load Type : = download,<br>Load Service Used : = Pull_Segment<br>Terminate Reason contained in the request |

#### 6.2.6.4.3.4 Download state transition table

Table 68 gives the download state transitions.

**Table 68 – Pull download state table**

| Transition | Current State | Event/Action | Next State |
|---|---|---|---|
| 1 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload State = UPLOADABLE<br><br>Initiate_Load.Rsp(+)<br>Pull_Segment.Req | DOWNLOADING |
| 2 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload State <>UPLOADABLE<br><br>Initiate_Load.Rsp(-)<br>Error Info : = object constraint conflict | DOWNLOADABLE |
| 3 | DOWNLOADING | Pull_Segment.Cnf(+)<br>&&More Follows=TRUE<br><br>Store received data<br>Pull_Segment.Req | DOWNLOADING |
| 4 | DOWNLOADING | Pull_Segment.Cnf(+)<br>&&More Follows=FALSE<br><br>Store received data<br>Terminate_Load.Req<br>Terminate Reason : = success | DOWNLOAD SUCCESS |
| 5 | DOWNLOADING | Pull_Segment.Cnf(-)<br><br>Discard received data<br>Terminate_Load.Req<br>Terminate Reason : = failure | DOWNLOAD FAILURE |
| 6 | DOWNLOADING,<br>DOWNLOAD FAILURE,<br>DOWNLOAD SUCCESS | Abort.Ind<br><br>Discard received data | DOWNLOADABLE |
| 7 | DOWNLOAD FAILURE | Terminate_Load.Cnf(+/-)<br><br>Discard received data | DOWNLOADABLE |
| 8 | DOWNLOAD SUCCESS | Terminate_Load.Cnf(-)<br><br>Discard received data | DOWNLOADABLE |
| 9 | DOWNLOAD SUCCESS | Terminate_Load.Cnf(+)<br><br>Update Contents Size attribute if present | LOADED |
| 10 | LOADED | Initiate_Load.Ind<br>&&Upload state attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Initiate_Load.Rsp(+) | DOWNLOADING |
| 11 | LOADED | Discard.Ind<br>&&Upload State attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Discard.Rsp(+) | DOWNLOADABLE |
| 12 | IN USE | Number of Related object In Use attribute incremented<br>&&its new value > 1<br><br>Set Related Object In Use flag value to TRUE if present (Start/Resume) | IN USE |
| 13 | IN USE | Number of Related Object In Use attribute decremented<br>&&its new value >0<br><br>Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill) | IN USE |

| Transition | Current State | Event/Action | Next State |
|---|---|---|---|
| 14 | LOADED | Initiate_Load.Ind<br>&&Upload State attribute in state <> UPLOADABLE<br>II Service fails(e.g. non erasable load region contents)<br><br>Initiate_Load.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 15 | LOADED | Discard.Ind<br>&& Upload State attribute in state <> UPLOADABLE<br>II Service fails(e.g. non erasable load region contents)<br><br>Discard.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 16 | LOADED | Number of Related Object In Use attribute incremented<br>&&its new value=1<br><br>Set Related Object In Use flag value to TRUE if any<br>(Start/Stop, Related objects are FI) | IN USE |
| 17 | IN USE | Number of Related Object In Use Attribute decremented<br>&&its new value=0<br><br>Set Related Object In Use flag value to FALSE if any<br>(Reset/Stop/Kill, Related objects are FI) | LOADED |

#### 6.2.6.4.4 Push download state machine

##### 6.2.6.4.4.1 Description

This kind of transfer uses the following services: Initiate Load, Pull Segment and Terminate Load

A Server may initiate a downloading transfer. In this case it issues an Initiate_Load.Req service primitive to a Client. This primitive tells the Client to initiate downloading transfer for a specified Load Region object. Once the Client has pushed the contents, it issues an Initiate_Load.Rsp(+/-) to the Server.

A Client may initiate a downloading transfer. In this case it issues an Initiate_Load.Req service primitive to a Server. This primitive tells the Server that the contents of a specified load region object will be pushed. Once the Client has pushed the contents, it ends the transfer by issuing a Terminate_Load.req to the Server

##### 6.2.6.4.4.2 Sequencing of service primitives

Table 69 summarizes the sequencing of service primitives.

**Table 69 – Push download sequencing of service primitives**

| FAL User(a Client) Owner of a calling Object | | FAL User(a Server) Owner of called Object | |
|---|---|---|---|
| Initiate_Load.Ind | ←----(T 1)------ | Initiate_Load.Req | |
| Initiate_Load.Req | -------(T2)-----→ | Initiate_Load.Ind | (Transfer started) |
| Initiate_Load.Cnf | ←----(T2)------- | Initiate_Load.Rsp | |
| Push_Segment.Req | -------(T3)-----→ | Push_Segment.Ind | (Transfer continued) |
| Push_Segment.Cnf | ←----(T3)------- | Push_Segment.Rsp | |
| | | | |
| Terminate_Load.Req | -------(T4)-----→ | Terminate_Load.Ind | (Transfer ended) |
| Terminate_Load.Cnf | ←----(T4)------- | Terminate_Load.Rsp | |
| Initiate_Load.Rsp | -------(T1)-----→ | Initiate_Load.Cnf | |

#### 6.2.6.4.4.3 Service parameter constraints

Table 70 summarizes the service parameter constraints.

**Table 70 – Push download service parameter constraints**

| Transition Number | Parameter constraints |
|---|---|
| T1 | Initiate_Load:<br>Calling Load Region Key Attribute selected,<br>Load Type : = download,<br>Service To Use : = Push_Segment,<br>Load Service Initiator : = FALSE |
| T2 | Initiate_Load<br>Called Load Region Key Attribute selected,<br>Load Type : = upload<br>Service To Use : = Push_Segment<br>Load Service Initiator : = TRUE |
| T3 | Pull_Segment<br>Load Type : = upload,<br>Segment Number : = ordered 1 to n if any<br>More Follows : = TRUE/FALSE with FALSE for the last segment |
| T4 | Terminate_Load<br>Load Type : = upload,<br>Load Service Used : = Pull_Segment<br>Terminate Reason contained in the response |

#### 6.2.6.4.4.4 Download state transition table

Table 71 gives the upload state transitions.

**Table 71 – Push download state table**

| Transition | Current State | Event | Next State |
|---|---|---|---|
| 1 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload state attribute in state UPLOADABLE<br><br>Initiate_Load.Rsp(+) | DOWNLOADING |
| 2 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload state attribute in state <>UPLOADABLE<br><br>Initiate_Load.Rsp(-)<br>with Error Info : = object constraint conflict | DOWNLOADABLE |
| 3 | DOWNLOADING | Push_Segment.Ind<br>&&More Follows=TRUE<br>&&compliance with Inter Segment Req Time Out constraint<br><br>Push_Segment.Rsp(+) | DOWNLOADING |
| 4 | DOWNLOADING | Push_Segment.Ind<br>&&More Follows=FALSE<br>&&compliance with Inter Segment Req Time Out constraint<br><br>Push_Segment.Rsp(+) | DOWNLOAD SUCCESS |
| 5<br>exist if supports Inter Segment Req Time Out | DOWNLOADING | Push_Segment.Ind<br>&&More Follows=TRUE II FALSE<br>&&no compliance with Inter Segment Req Time Out constraint<br><br>Push.Segment.Rsp(-)<br>Discard received data | DOWNLOAD FAILURE |
| 6 | DOWNLOADING,<br>DOWNLOAD FAILURE,<br>DOWNLOAD SUCCESS | Abort.Ind<br><br>Discard received data | DOWNLOADABLE |
| 7 | DOWNLOAD FAILURE | Terminate_Load.Ind<br><br>Terminate_Load.Rsp(+/-)<br>Terminate Reason : = success/failure | DOWNLOADABLE |
| 8 | DOWNLOAD SUCCESS | Terminate_Load.Ind<br>&&service fails<br><br>Terminate_Load.Rsp(-)<br>Terminate Reason : = failure<br>Discard received data | DOWNLOADABLE |
| 9 | DOWNLOAD SUCCESS | Terminate_Load.Ind<br>&&service succeeds(e.g. all segments received)<br><br>Update Contents Size attribute if present<br>Terminate_Load.Rsp(+)<br>Terminate Reason : = success | LOADED |
| 10 | LOADED | Initiate_Load.Ind<br>&&Upload State attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Initiate_Load.Rsp(+) | DOWNLOADING |
| 11 | LOADED | Discard.Ind<br>&&Upload State attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Discard.Rsp(+) | DOWNLOADABLE |
| 12 | IN USE | Number of Related object In Use attribute incremented<br>&&its new value > 1 | IN USE |

| Transition | Current State | Event | Next State |
|---|---|---|---|
| | | Set Related Object In Use flag value to TRUE if any (Start/Resume) | |
| 13 | IN USE | Number of Related Object In Use attribute decremented<br>&&its new value >0<br><br>Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill) | IN USE |
| 14 | LOADED | Initiate_Load.Ind<br>&&Upload State attribute in state <> UPLOADABLE<br>II Service fails(e.g. non erasable load region contents)<br><br>Initiate_Load.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 15 | LOADED | Discard.Ind<br>&&Upload State attribute in state <> UPLOADABLE<br>\|\| Service fails(e.g. non erasable load region contents)<br><br>Discard.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 16 | LOADED | Number of Related Object In Use attribute incremented<br>&&its new value=1<br><br>Set Related Object In Use flag value to TRUE if any (Start/Stop, Related objects are FI) | IN USE |
| 17 | IN USE | Number of Related Object In Use Attribute decremented<br>&&its new value=0<br><br>Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill, Related objects are FI) | LOADED |

### 6.2.7 Function invocation ASE

#### 6.2.7.1 Overview

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

Stateless function invocations, referred to as actions, run to completion when invoked and cannot be interrupted. In addition, some atomic actions return a value in response to being invoked, while others do not. Those that do may be used to model software *functions*, and those that do not may be used to model software *procedures*.

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

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

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

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

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

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

#### 6.2.7.2 Function invocation model class specifications

##### 6.2.7.2.1 Function invocation class specification

###### 6.2.7.2.1.1 Class overview

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

###### 6.2.7.2.1.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **FUNCTION INVOCATION ASE** |
| **CLASS:** | | | FUNCTION INVOCATION |
| **CLASS ID:** | | | 3 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Access Privilege |
| 1.1 | (m) | Attribute: | Password |
| 1.2 | (m) | Attribute: | Access Groups |
| 1.3 | (m) | Attribute: | Access Rights |
| 2 | (m) | Attribute: | Deletable (TRUE,FALSE) |
| 1 | (m) | Attribute: | Reusable (TRUE,FALSE) Default : TRUE |
| 2 | (m) | Attribute: | Function Invocation State |
| 4 | (m) | Attribute: | Number of Related Objects In Use |
| 3 | (o) | Attribute: | List of Related Objects |
| 3.1 | (o) | Attribute: | ClassID |
| 3.2 | (o) | Attribute: | Numeric ID |
| 5.2 | (m) | Attribute: | In Use Flag (TRUE, FALSE) |
| 4 | (c) | Constraint: | Start Service Supported |
| 4.1 | (o) | Attribute: | Start Service Argument Data type |
| 5 | (c) | Constraint: | Stop Service Supported |
| 5.1 | (o) | Attribute: | Stop Service Argument Data type |
| 6 | (c) | Constraint: | Resume Service Supported |
| 6.1 | (o) | Attribute: | Resume Service Argument Data type |
| 7 | (c) | Constraint: | Reset Service Supported |
| 7.1 | (o) | Attribute: | Reset Service Argument Data type |
| 8 | (c) | Constraint: | Kill Service Supported |
| 8.1 | (o) | Attribute: | Kill Service Argument Data type |
| 9 | (o) | Attribute: | Last Execution Argument |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Start |
| 2 | (o) | OpsService: | Stop |
| 3 | (o) | OpsService: | Resume |
| 4 | (o) | OpsService: | Reset |
| 5 | (o) | OpsService: | Kill |

###### 6.2.7.2.1.3 Attributes

**Access Privilege**

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

**Password**

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

**Access Groups**

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

**Access Rights**

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

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

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

**Deletable**

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

**Reusable**

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

**Function Invocation State**

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

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

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

RUNNING This state indicates that the function invocation is executing.

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

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

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

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

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

**Number of Related Objects in Use**

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

**List of Related Objects**

This optional attribute specifies the Load Region objects related to this function invocation.

**Class ID**

This attribute is the class identifier for the related object.

**Numeric ID**

This attribute is the numeric identifier for the related object.

**In-use flag**

This attribute indicates, when TRUE, that the related object is currently in the IN USE state. This attribute is updated each time the related Load Region Object state attribute goes/leaves the IN USE state. The Load Region State Machine gives the constraints for these transitions

**Start Service Argument Data type**

This attribute identifies the data type for each of the execution arguments for the start service supported by this function invocation. This attribute is present if the function invocation supports the start service. If the start service is supported but has no execution argument, the value of this attribute is NULL.

**Stop Service Argument Data type**

This attribute identifies the data type for the execution argument for the stop service supported by this function invocation. This attribute is present if the function invocation supports the stop service. If the stop service is supported but has no execution argument, the value of this attribute is NULL.

**Resume Service Argument Data type**

This attribute identifies the data type for the execution argument for the resume service supported by this function invocation. This attribute is present if the function invocation supports the resume service. If the resume service is supported but has no execution argument, the value of this attribute is NULL.

**Reset Service Argument Data type**

This attribute identifies the data type for the execution argument for the reset service supported by this function invocation. This attribute is present if the function invocation supports the reset service. If the reset service is supported but has no execution argument, the value of this attribute is NULL.

**Kill Service Argument Data type**

This attribute identifies the data type for the execution argument for the kill service supported by this function invocation. This attribute is present if the function invocation supports the kill service. If the kill service is supported but has no execution argument, the value of this attribute is NULL.

**Last Execution Argument**

This optional attribute specifies the value for the Execution Argument that was last sent by the Client.

#### 6.2.7.2.1.4 Services

**Start**

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

**Stop**

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

**Resume**

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

**Reset**

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

**Kill**

This optional service is used to request that the current execution of the function invocation be aborted so that it may not be resumed. Once killed, the context of the function execution is lost.

### 6.2.7.2.2 Action class specification

#### 6.2.7.2.2.1 Class overview

The Action Class is used to define stateless function invocations. Stateless function invocations run to completion when invoked. Their execution may not be aborted using the services of the FAL. They may return a result using the Return service.

#### 6.2.7.2.2.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **FUNCTION INVOCATION ASE** |
| **CLASS:** | | ACTION | |
| **CLASS ID:** | | | 19 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Reusable (TRUE,FALSE) Default : TRUE |
| 2 | (o) | Attribute: | List of Related Objects |
| 3.1 | (m) | Attribute: | ClassID |
| 3.2 | (m) | Attribute: | Numeric ID |

| | | | |
|---|---|---|---|
| 3.2 | (m) | Attribute: | In Use Flag (TRUE, FALSE) |
| 4 | (o) | Attribute: | Action Invoke Service Argument Data type |
| 5 | (c) | Constraint: | Action Return Service Supported |
| 5.1 | (o) | Attribute: | Action Return Service Argument Data type |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (m) | OpsService: | Action Invoke |
| 2 | (o) | OpsService: | Action Return |

#### 6.2.7.2.2.3 Attributes

**Reusable**

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

**List of Related Objects**

This optional attribute specifies the Load Region objects related to this function invocation.

**Class ID**

This attribute is the class identifier for the related object.

**Numeric ID**

This attribute is the numeric identifier for the related object.

**In-use Flag**

This attribute indicates, when TRUE, that the related object is currently in the IN USE state. This attribute is updated each time the Load Region Object State attribute goes/leaves the IN USE state. The Load Region State Machine gives the constraints for these transitions.

**Action Invoke Service Argument Data type**

This attribute identifies the data type for the execution argument for the action invoke service supported by this action. This attribute is present if the action supports the action invoke service. If the action invoke service is supported but has no execution argument, the value of this attribute is NULL.

**Action Return Service Argument Data type**

This attribute identifies the data type for the execution argument for the action invoke service supported by this action. This attribute is present if the action supports the action invoke service. If the action invoke service is supported but has no execution argument, the value of this attribute is NULL.

#### 6.2.7.2.2.4 Services

**Action Invoke**

This mandatory service is used to call the action object with an argument list.

**Action Return**

This optional service is used by the action object to return the results of its invocation.

### 6.2.7.3 Function invocation model service specifications

#### 6.2.7.3.1 Supported services

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

Start

Stop

Resume
Reset
Kill
Action Invoke
Action Return

#### 6.2.7.3.2 Start service

##### 6.2.7.3.2.1 Service overview

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

##### 6.2.7.3.2.2 Service primitives

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

**Table 72 – Start service parameters**

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

NOTE See the Note in 3.8.4.3.

**Argument**

This parameter carries the parameters of the service invocation.

**Key attribute**

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

**Execution Argument**

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data types for the function invocation object.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Function Invocation State**

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

#### 6.2.7.3.2.3 Service procedure

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

### 6.2.7.3.3 Stop service

#### 6.2.7.3.3.1 Service overview

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

#### 6.2.7.3.3.2 Service primitives

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

**Table 73 – Stop service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

This parameter carries the parameters of the service invocation.

**Key Attribute**

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

**Execution Argument**

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data types for the function invocation object.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Function Invocation State**

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

#### 6.2.7.3.3.3 Service procedure

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

### 6.2.7.3.4 Resume service

#### 6.2.7.3.4.1 Service overview

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

#### 6.2.7.3.4.2 Service primitives

The service parameters for this service are shown in Table 74.

**Table 74 – Resume service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

This parameter carries the parameters of the service invocation.

**Key Attribute**

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

**Execution Argument**

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data types for the function invocation object.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Function Invocation State**

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

#### 6.2.7.3.4.3 Service procedure

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

### 6.2.7.3.5 Reset service

#### 6.2.7.3.5.1 Service overview

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

#### 6.2.7.3.5.2 Service primitives

The service parameters for this service are shown in Table 75.

**Table 75 – Reset service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

This parameter carries the parameters of the service invocation.

**Key Attribute**

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

**Execution Argument**

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data types for the function invocation object.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Function Invocation State**

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

#### 6.2.7.3.5.3 Service procedure

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

### 6.2.7.3.6 Kill service

#### 6.2.7.3.6.1 Service overview

This confirmed service is used to abort the execution of a function invocation so that it may not be restarted or resumed.

#### 6.2.7.3.6.2 Service primitives

The service parameters for this service are shown in Table 76.

**Table 76 – Kill service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE See the Note in 3.8.4.3. | | | | |

**Argument**

This parameter carries the parameters of the service invocation.

**Key Attribute**

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

**Execution Argument**

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data types for the function invocation object.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Function Invocation State**

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

#### 6.2.7.3.6.3 Service procedure

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

### 6.2.7.3.7 Action invoke service

#### 6.2.7.3.7.1 Service overview

This unconfirmed service is used to invoke the execution of an action object.

#### 6.2.7.3.7.2 Service primitives

The service parameters for this service are shown in Table 77.

**Table 77 – Action invoke service parameters**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Key Attribute | M | M (=) |
| Action Invocation ID | M | M (=) |
| Execution Argument | C | C (=) |
| Timeliness | | C |
| Duplicate FAL PDU Body | | C |

**Argument**

This parameter carries the parameters of the service invocation.

**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 Action Invoke 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 Action Invoke is to be sent.

**Key Attribute**

This parameter specifies one of the key attributes of the action object.

**Action Invocation ID**

This parameter identifies a specific invocation of the action object. It permits correlation of the invocation and its result as reported with a Return service.

**Execution Argument**

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data types for the action object.

**Timeliness**

This conditional parameter indicates the data-link layer timeliness status for the transfer of this service request. This parameter is present if it is supported on the specified AR.

**Duplicate FAL PDU Body**

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the data-link layer. It is present if supported in the DL Mapping Attributes of the AREP.

#### 6.2.7.3.7.3 Service procedure

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

### 6.2.7.3.8 Action return service

#### 6.2.7.3.8.1 Service overview

This unconfirmed service is used return the results from an invocation of an action object.

#### 6.2.7.3.8.2 Service primitives

The service parameters for this service are shown in Table 78.

**Table 78 – Action return service parameters**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Action Invocation ID | M | M (=) |
| Action Success | S | S (=) |
| Return Value | U | U (=) |
| Action Failure | S | S (=) |
| Error Info | M | M (=) |

**Argument**

This parameter carries the parameters of the service invocation.

**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 Action Return 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 Action Return is to be sent.

**Action Invocation ID**

This parameter identifies the specific invocation of the action object whose results are being returned.

**Action Success**

This selection type parameter is present if the action was successful.

**Return Value**

This optional parameter carries the results of this invocation of the action.

**Action Failure**

This selection type argument is present if the action was not successful.

#### 6.2.7.3.8.3 Service procedure

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

The AP containing the action submits an Action Return Service request primitive to return the results of a previous invocation of the action. An action invocation identifier is included in the request to identify the particular invocation. This identifier is used only by the receiving user, and is only conveyed by the FAL

### 6.2.7.4 Function invocation state machine

Table 79 gives the State Transitions for a Function Invocation object. The state “NON EXISTENT” indicates that the object does not exist. It is not defined as a valid value for the function invocation state attribute because attributes are created and deleted with the object (and therefore have no value when the object does not exist).

**Table 79 – State transitions for a function invocation object**

| Transition | Current State | Events | Next State |
|---|---|---|---|
| 1 | NON EXISTENT | Create.Ind<br>&& List of Attribute IDs and Initial Values parameter = List of Load Region Object IDs<br>&& Specified Load Region objects available<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Create.Rsp(+) | Idle |
| 2 | NON EXISTENT | Create.Ind<br>&& List of Attribute IDs and Initial Values parameter <> List of Load Region Object IDs<br>\|\| Specified Load Region objects not available<br><br>Create.Rsp(-) | NON EXISTENT |
| 3 | IDLE | Delete.Ind<br>&& Deleteable=TRUE<br><br>Delete.Rsp(+) | NON EXISTENT |
| 4 | IDLE | Delete.Ind<br>&& Deleteable=FALSE<br><br>Delete.Rsp(-) | IDLE |
| 5 | IDLE | Start.Ind | STARTING |
| x | STARTING | Start succeeds: e.g.<br>Related Load Region Object in state LOADED or IN-USE<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Start.Rsp(+) | RUNNING |
| 7 | STARTING | Start fails, non destructive: e.g.<br>Related Load Region Object not in state LOADED or IN USE<br><br>Start.Rsp(-) | IDLE |
| 8 | STARTING | Start fails, destructive<br><br>Start.Rsp(-) | UN-RUNNABLE |
| 9 | RUNNING | Stop.Ind | STOPPING |
| 10 | STOPPING | Stop succeeds: e.g.<br>Related Load Region Object in state IN USE<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Stop.Rsp(+) | STOPPED |
| 11 | STOPPING | Stop Execution fails, non destructive<br><br>Stop.Rsp(-) | RUNNING |

| Transition | Current State | Events | Next State |
|---|---|---|---|
| 12 | STOPPING | Stop fails, destructive<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Stop.Rsp(-) | UN-RUNNABLE |
| 13 | STOPPED | Resume.Ind | RESUMING |
| 14 | RESUMING | Resume succeeds: e.g.<br>Related Load Region Object in state Loaded or In-Use<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Resume.Rsp(+) | RUNNING |
| 15 | RESUMING | Resume fails, non destructive: e.g.<br>Load Region not in state LOADED or IN-USE<br><br>Resume.Rsp(-) | STOPPED |
| 16 | RESUMING | Resume fails, destructive<br><br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Resume.Rsp(-) | UNRUNNABLE |
| 17 | STOPPED | Reset.Ind | RESETTING |
| 18 | RESETTING | Reset succeeds, reusable=TRUE<br><br>Reset.Rsp(+) | IDLE |
| 19 | RESETTING | Reset succeeds, reusable = false<br><br>Reset.Rsp(+) | UNRUNNABLE |
| 20 | RESETTING | Reset fails, non destructive<br><br>Reset.Rsp(-) | STOPPED |
| 21 | RESETTING | Reset fails, destructive<br><br>Reset.Rsp(-) | UNRUNNABLE |
| 22 | RESETTING | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 23 | STOPPING | Kill.Ind<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Kill.Rsp(+) | UNRUNNABLE |

| Transition | Current State | Events | Next State |
|---|---|---|---|
| 24 | RUNNING | End of Program<br>&& Reusable=TRUE<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported) | IDLE |
| 25 | RUNNING | End of Program<br>&& Reusable = false<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported) | UNRUNNABLE |
| 26 | RUNNING | Program stopped<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported) | STOPPED |
| 27 | STOPPED | Delete.Ind<br>&& this Function Invocation is not deletable<br><br>Delete.Rsp(-) | STOPPED |
| 28 | STOPPED | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 29 | STOPPED | Delete.Ind<br>&& this Function Invocation is deletable<br><br>Delete.Rsp(+) | NON EXISTENT |
| 30 | RESUMING | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 31 | RUNNING | Kill.Ind<br><br>For the Function Invocation object, update the attributes:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>For each related Load Region Object, update the attributes:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Kill.Rsp(+) | UNRUNNABLE |
| 32 | STARTING | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 33 | IDLE | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 34 | UNRUNNABLE | Delete.Ind<br><br>Delete.Rsp(+) | NON EXISTENT |

## 6.3 ARs

### 6.3.1 Queued user-triggered unidirectional (QUU) AR endpoint class specification

#### 6.3.1.1 Class overview

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

An AR ASE user wishing to convey a request APDU submits an AR ASE Service Data Unit to the sending endpoint of the AR. The AREP sending the request APDU submits it to its underlying layer for transfer. The underlying layer sends it at its next opportunity. The AREP receiving the request APDU from its underlying layer delivers it to the AR ASE user in the order that it was received.

The following summarizes the characteristics of this AREP class.

| | |
|---|---|
| Roles: | REPORT SOURCE<br>REPORT SINK |
| Cardinality: | 1-to-n |
| Timeliness: | No |

#### 6.3.1.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | Queued User-triggered Uni-directional AR Endpoint | | |
| **CLASS ID:** | | | 36 |
| **PARENT CLASS:** | | | AR Endpoint |
| **NETWORK MANAGEMENT ATTRIBUTES:** | | | |
| 1. | (m) | Attribute: | Role (REPORT SOURCE, REPORT SINK) |
| 2. | (m) | Attribute: | AREP State (OPEN, CLOSED) |
| 3. | (m) | Attribute: | Remote Address Configuration Type (FREE, LINKED) |
| 4. | (m) | Attribute: | DL Mapping Reference |
| **SERVICES:** | | | |
| 1. | (o) | OpsService: | Unconfirmed Send |

#### 6.3.1.3 Network management attributes

**Role**

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

REPORT SOURCE Endpoints of this type distribute reports to multiple endpoints using unconfirmed services operating over connectionless data link services.

REPORT SINK Endpoints of this type receive reports from source endpoints using unconfirmed services operating over connectionless data link services.

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Remote Address Configuration Type**

This attribute specifies how the remote address of the AREP is configured.

FREE The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the Unconfirmed service request.

LINKED The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6-5).

**DL Mapping Reference**

This attribute is a reference to the underlying data-link layer mapping. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

#### 6.3.1.4 Services

**Unconfirmed Send**

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

### 6.3.2 Queued user-triggered bi-directional connection-oriented (QUB-Co) AR endpoint class specification

#### 6.3.2.1 Class overview

This class is defined to support the on-demand exchange of confirmed services between two application processes. Unconfirmed services are not supported by this type of AR. It uses connection-oriented data link services for the exchanges. The behavior of these classes is described as follows.

An AR ASE user wishing to convey a request APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

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

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

The AREP that issued the request APDU receives the response APDU from its underlying layer and queues it for delivery to its AR ASE user in the order that it was received. It also stops its associated service response timer.

The following summarizes the characteristics of this AREP class.

| | |
|---|---|
| Roles: | Client<br>Server<br>Peer |
| Cardinality: | 1-to-1 |
| Timeliness: | No |

#### 6.3.2.2 Formal model

| | |
|---|---|
| **FAL ASE:** | **AR ASE** |
| **CLASS:** | Queued User-triggered Bi-directional Connection-Oriented AREP |
| **CLASS ID:** | 34 |
| **PARENT CLASS:** | AR Endpoint |

**NETWORK MANAGEMENT ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1. | (m) | Attribute: | Role (CLIENT, SERVER, PEER) |
| 2. | (m) | Attribute: | AREP State |
| 3. | (m) | Attribute: | Remote Address Configuration Type (FREE, LINKED) |
| 4. | (m) | Attribute: | Maximum Outstanding Requests Calling |
| 5. | (m) | Attribute: | Maximum Outstanding Requests Called |
| 6. | (m) | Attribute: | Initiator (TRUE, FALSE) |
| 7. | (o) | Attribute: | Remote AP Name |
| 8. | (m) | Attribute: | Transmit DL Mapping Reference |
| 9. | (m) | Attribute: | Receive DL Mapping Reference |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1. | (o) | OpsService: | Confirmed Send |
| 2. | (o) | OpsService: | Establish |

#### 6.3.2.3 Network management attributes

**Role**

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

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

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

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

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Remote Address Configuration Type**

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

FREE The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the Unconfirmed service request.

LINKED The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6-5).

**Max Outstanding Requests Calling**

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

**Max Outstanding Requests Called**

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

**Initiator**

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

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

**Remote AP Name**

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

**Transmit DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**Receive DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the receive conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

#### 6.3.2.4 Services

**Confirmed Send**

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

**Establish**

This service is used to establish an AR.

### 6.3.3 Queued user-triggered bi-directional connectionless (QUB-CL) AR endpoint class specification

#### 6.3.3.1 Class overview

This class is defined to support the on-demand exchange of confirmed and unconfirmed services between two or more application processes. This class uses connectionless data link services for the exchanges. The data-link layer transfers may be either acknowledged or unacknowledged. The behavior of this class is described as follows.

An AR ASE user wishing to convey a request APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

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

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

The AREP that issued the request APDU receives the response APDU from its underlying layer and queues it for delivery to its AR ASE user in the order that it was received. It also stops its associated service response timer.

The following summarizes the characteristics of this AREP class.

| | |
|---|---|
| Roles: | Client<br>Server<br>Peer |
| Cardinality: | 1-to-n |
| Timeliness: | No |

#### 6.3.3.2 Formal model

**FAL ASE:** **AR ASE**
**CLASS:** Queued User-triggered Bi-directional Connectionless AREP
**CLASS ID:** 45
**PARENT CLASS:** AR Endpoint
**NETWORK MANAGEMENT ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | (m) | Attribute: | Role (CLIENT, SERVER, PEER) |
| 2 | (m) | Attribute: | AREP State |
| 3 | (m) | Attribute: | Remote Address Configuration Type (FREE, LINKED) |
| 4 | (m) | Attribute: | Maximum Outstanding Requests Calling |
| 5 | (m) | Attribute: | Maximum Outstanding Requests Called |
| 6 | (m) | Attribute: | Initiator (TRUE, FALSE) |
| 7 | (o) | Attribute: | Remote AP Name |
| 8 | (m) | Attribute: | Transmit DL Mapping Reference |
| 9 | (m) | Attribute: | Receive DL Mapping Reference |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Confirmed Send |
| 2 | (o) | OpsService: | Unconfirmed Send |
| 3 | (o) | OpsService: | Establish |
| 4 | (o) | OpsService: | Abort |

#### 6.3.3.3 Network management attributes

**Role**

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

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

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

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

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Remote Address Configuration Type**

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

FREE The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically.

LINKED The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6-5).

**Max Outstanding Requests Calling**

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

**Max Outstanding Requests Called**

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

**Initiator**

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR should be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

**Remote AP Name**

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

**Transmit DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**Receive DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the receive conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

#### 6.3.3.4 Services

**Confirmed Send**

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

**Unconfirmed Send**

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

**Establish**

This service is used to establish an AR.

**Abort**

This service is used to abort an AR.

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

#### 6.3.4.1 Class overview

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

The behavior of this class is described as follows.

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

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

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

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

Additionally the AREP monitors the connection in cases if no service from the AR ASE user is outstanding by sending an IdlePDU to the peer instance. The TCTimer (Transmit Client Timer) and the TSTimer (Transmit Server Timer) monitor the sending of IdlePDUs on both connection endpoints. The RCTimer (Receive Client Timer) and RSTimer (Receive Server Timer) monitor the liveness of the connection on both connection endpoints. The RCTimer or RSTimer expire if no PDU (also no IdlePDU) has been received during a certain amount of time and causes the termination of the AREP connection.

Furthermore, the AR ASE user at server site is able to interrupt or continue the data flow by issuing the AR-XON-XOFF service.

The following summarizes the characteristics of this AREP class.

| | |
|---|---|
| Roles | Client<br>Server<br>Peer |
| Cardinality | 1-to-1 |
| Timeliness | No |

#### 6.3.4.2 Formal model

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

| | | | |
|---|---|---|---|
| 3 | (o) | OpsService: | Establish |
| 4 | (o) | OpsService: | Abort |

#### 6.3.4.3 Network management attributes

**Role**

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

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

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

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

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Remote Address Configuration Type**

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

FREE The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the Unconfirmed service request.

LINKED The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6-5).

**Max outstanding Requests Calling**

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

**Max Outstanding Requests Called**

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

**Max Outstanding Unconfirmed Requests Client (maxUCSC)**

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

**Max Outstanding Unconfirmed Requests Server (maxUCSS)**

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

**Maximum Outstanding Requests Client (maxOSCC)**

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

**Maximum Outstanding Requests Server (maxOSCS)**

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

**Initiator**

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR should be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

**Remote AP Name**

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

**Transmit DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**Receive DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the receive conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**CIU**

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

#### 6.3.4.4 Services

**Confirmed Send**

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

**Establish**

This service is used to establish an AR.

**Abort**

This service is used to disconnect an AR.

**Unconfirmed Send**

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

### 6.3.5 Queued user-triggered bi-directional with segmentation(QUB-Seg) AR endpoint class specification

#### 6.3.5.1 Class overview

This class is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. It uses connection-oriented datalink services for the exchanges. It supports segmentation of APDUs within the Application Layer. The data link priority for the transfers is specified separately for each transfer direction.

An AR ASE service data unit (if its size is greater than a single transferable element) can be split into several APDUs which are sent to its underlying layer to be transferred. The underlying layer sends each segment at the next opportunity. The AREP receiving the request APDU segments from its underlying layer reassembles the APDU and delivers it to the AR ASE. Delivery of multiple APDUs is done in the order in which they are completely received.

The following summarizes the characteristics of this AREP class.

| | |
|---|---|
| Roles: | Client<br>Server<br>Peer |
| Cardinality: | 1-to-1 |
| Timeliness: | No |

#### 6.3.5.2 Formal model

| | |
|---|---|
| **FAL ASE:** | AR ASE |
| **CLASS:** | Queued User-triggered Bi-directional Segmentation AREP |
| **CLASS ID:** | 45 |
| **PARENT CLASS:** | AR Endpoint |

**NETWORK MANAGEMENT ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1. | (m) | Attribute: | Role (CLIENT, SERVER, PEER) |
| 2. | (m) | Attribute: | AREP State |
| 3. | (m) | Attribute: | Remote Address Configuration Type (FREE, LINKED) |
| 4. | (m) | Attribute: | Maximum Outstanding Requests Calling |
| 5. | (m) | Attribute: | Maximum Outstanding Requests Called |
| 6. | (m) | Attribute: | Initiator (TRUE, FALSE) |
| 7. | (o) | Attribute: | Remote AP Name |
| 8. | (m) | Attribute: | Transmit DL Mapping Reference |
| 9. | (m) | Attribute: | Receive DL Mapping Reference |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1. | (o) | OpsService: | Confirmed Send |
| 2. | (o) | OpsService: | Unconfirmed Send |
| 3. | (o) | OpsService: | Establish |
| 4. | (o) | OpsService: | Reject |
| 5. | (o) | OpsService: | Abort |

#### 6.3.5.3 Network management attributes

**Role**

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

> PEER (CLIENT/SERVER) Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type should indicate whether or not they are to be the initiator of the AR establishment process.
>
> CLIENT Endpoints of this type issue confirmed service Request-APDUs to servers and receive confirmed service Response-APDUs and unconfirmed service Request PDU.
>
> SERVER Endpoints of this type receive confirmed and unconfirmed service Request-APDUs from clients and issue confirmed service Response-APDUs and unconfirmed service Request PDU to them. They may also issue unconfirmed service Request APDUs to clients.

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Remote Address Configuration Type**

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

| | |
|---|---|
| FREE | The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically. |
| LINKED | The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6-5). |

**Max Outstanding Requests Calling**

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

**Max Outstanding Requests Called**

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

**Initiator**

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR should be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

**Remote AP Name**

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

**Transmit DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**Receive DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the receive conveyance path for this AREP. DL mappings for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

#### 6.3.5.4 Services

**Confirmed Send**

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

**Unconfirmed Send**

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

**Establish**

This service is used to establish an AR.

**Abort**

This service is used to Abort an AR.

**Reject**

This provider initiated service is used to signal the detection of a protocol error.

### 6.3.6 Buffered user-triggered bi-directional (BUB) AR endpoint class specification

#### 6.3.6.1 Class overview

This class is defined to support the on-demand exchange of confirmed and of unconfirmed services between two application processes using buffers.

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

An AR ASE user wishing to convey a request or response APDU submits it to its AREP as an AR ASE Service Data Unit. The AREP sending the request or response APDU writes it into the data-link layer buffer, completely replacing the existing contents of the buffer. The data-link layer transfers the buffer contents at the earliest transfer opportunity. The buffer may also be transferred whenever a request to transmit (AR Compel) is received, either locally or from the network.

The AREP receiving the request or response APDU from its underlying layer, buffers it for delivery. If the APDU is a response, the endpoint stops its associated timer.

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

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

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

The following summarizes the characteristics of this AREP class.

| | |
|---|---|
| Roles: | Client<br>Server<br>Peer |
| Cardinality: | 1-to-1 |
| Timeliness: | No |

#### 6.3.6.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | | | Buffered User-triggered Bi-directional AREP |
| **CLASS ID:** | | | 42 |
| **PARENT CLASS:** | | | AR Endpoint |
| **NETWORK MANAGEMENT ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Role (CLIENT, SERVER, PEER) |
| 2 | (m) | Attribute: | AREP State |
| 3 | (m) | Attribute: | Confirmed Services Flag(TRUE/FALSE) |
| 4 | (m) | Attribute: | Initiator (TRUE, FALSE) |
| 5 | (m) | Attribute: | Transmit DL Mapping Reference |
| 6 | (m) | Attribute: | Receive DL Mapping Reference |
| 7 | (c) | Constraint: | Confirmed Services = TRUE |
| 7.1 | (m) | Attribute: | Max Outstanding Requests Calling |

| | | | |
|---|---|---|---|
| 7.2 | (m) | Attribute: | Max Outstanding Requests Called |
| 8 | (m) | Attribute: | Remote AP Name |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Unconfirmed Send |
| 2 | (o) | OpsService: | AR Compel |
| 3 | (o) | OpsService: | Confirmed Send |
| 4 | (o) | OpsService: | Establish |
| 5 | (o) | OpsService: | Get Buffered Message |

#### 6.3.6.3 Network management attributes

**Role**

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

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

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

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

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Confirmed Services Flag**

This attribute specifies, when TRUE, that this endpoint supports the conveyance of confirmed service requests.

**Initiator**

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR should be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

**Transmit DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path for this AREP. DL-Mapping References for AREPs are specified in IEC 61158-6-5.

**Receive DL Mapping Reference**

This attribute provides the mapping to the underlying layer for the receive conveyance path for this AREP. DL-Mapping References for AREPs are specified in IEC 61158-6-5.

**Max Outstanding Requests Calling**

This conditional attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP. This attribute is present only if this endpoint supports confirmed services (confirmed services = TRUE).

**Max Outstanding Requests Called**

This conditional attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user. This attribute is present only if this endpoint supports confirmed services (Confirmed Services = TRUE).

**Remote AP Name**

This attribute specifies the AP attached to the remote AREP.

#### 6.3.6.4 Services

**Unconfirmed Send**

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

**AR Compel**

This optional service is used to cause an APDU locally or remotely buffered for transmission in the data-link layer to be transmitted at the next available opportunity.

**Confirmed Send**

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

**Establish**

This service is used to establish an AR.

**Get Buffered Message**

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

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

#### 6.3.7.1 Class overview

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

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

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

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

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

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

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

This type of AR also supports, as an option, the ability of the AR ASE to determine publishing, transmission, or reception timeliness. For publishing timeliness, the sending AR ASE will mark the buffer if it has not been updated within a specified update interval. For reception timeliness, the receiving AR ASE will mark the buffer if it has not been received within a specified reception interval. Both publishing and reception timeliness information are given. For transmission timeliness, the sending AR ASE determines if the buffer has been transmitted within a specified interval. If the transmission does not occur, the FAL user is notified and the buffer contents are updated to indicate the lack of transmission timeliness.

Residence timeliness is supported by this type of AREP. Figure 5 illustrates this type of timeliness.

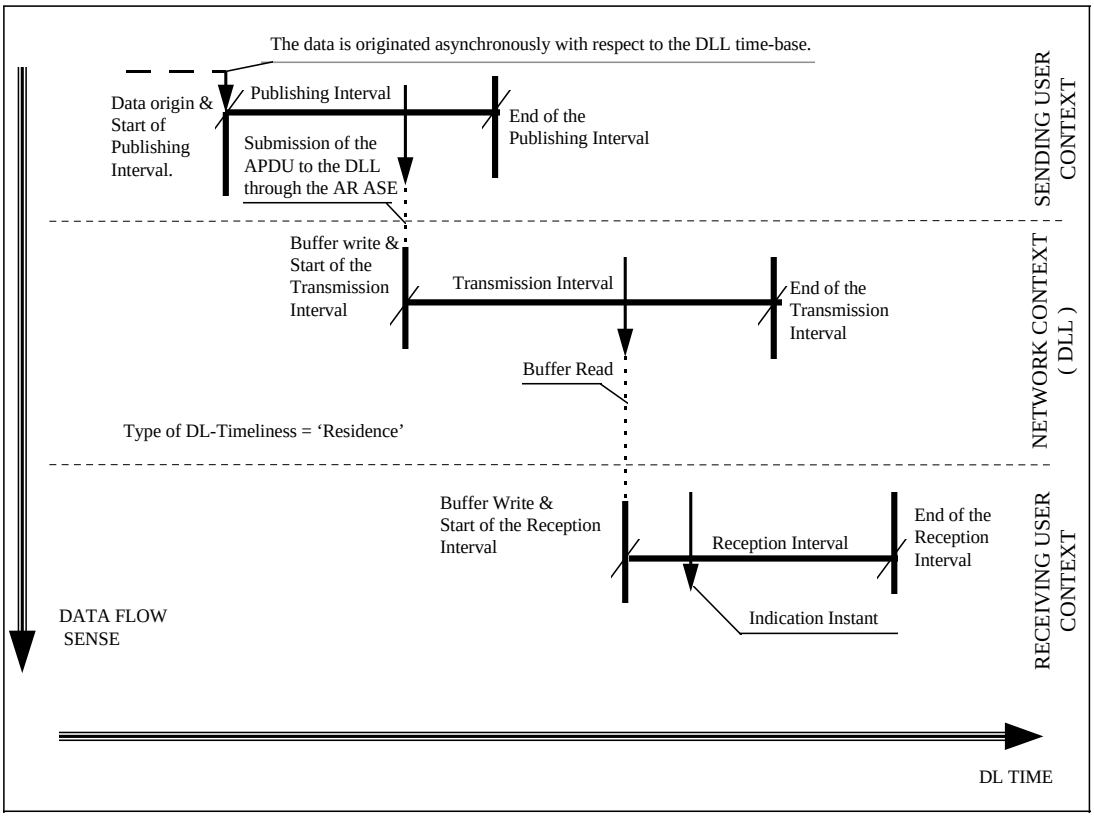



**Figure 5 – Residence timeliness**

This type of timeliness requires the use of a data link timer associated with data buffer (transmit or receive). The timer is started when the data is written into the transmit buffer by the publisher AREP or received from the network into the subscriber buffer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows:

Publisher AREP  Data is timely if it is transmitted within the timeliness window after it is placed into the data link buffer for transmission. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate its current status.

Subscriber AREP Data is timely if it is received as timely by the data-link layer and is read from the receive buffer by the AREP within the timeliness window that was started when the data was received. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent reads from the buffer of the same data will indicate its current status.

Synchronized timeliness is supported by this type of AREP. Figure 6 illustrates this type of timeliness.

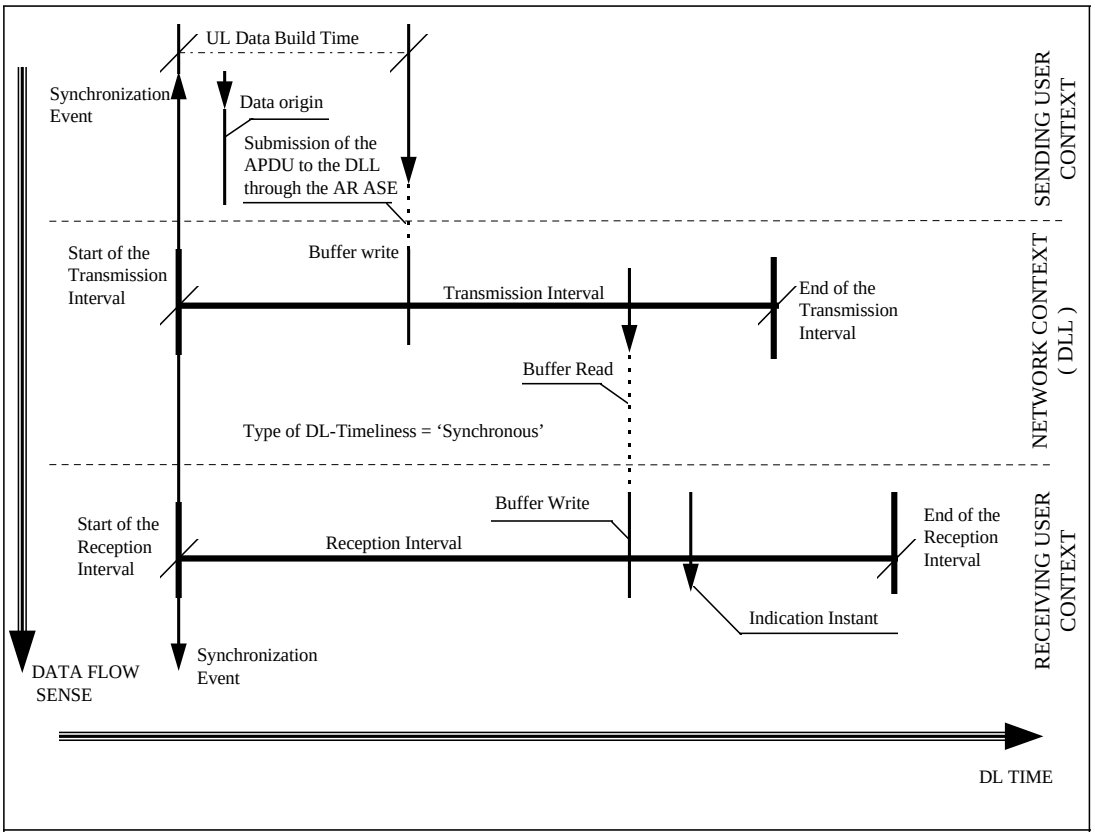



**Figure 6 – Synchronized timeliness**

This type of timeliness requires the use of two supporting data link connections. One is used to publish the data and the other is used to distribute a "sync mark" within the data-link layer to each of the stations participating as either a publisher or a subscriber. The sync mark causes a timer associated with data buffer (transmit or receive) to be started within the data-link layer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows:

Publisher AREP Data is timely if it is placed into the buffer within the time window after the receipt of a "sync mark". When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate the current status of the buffer.

Subscriber AREP Data is timely if it is received as timely by the data-link layer and is received within the time window. When the timeliness window expires, the data in the buffer is regarded as NOT Timely. Data received after the expiration of the window, but before the next sync mark is also considered NOT Timely. Subsequent reads from the buffer of the same data will indicate it current status.

The following summarizes the characteristics of this AREP class.

Roles: Push Publisher
Push Subscriber

Cardinality: 1-to-Many

Timeliness: Optional

#### 6.3.7.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | | | Buffered Network-Scheduled Uni-directional AR Endpoint |
| **CLASS ID:** | | | 38 |
| **PARENT CLASS:** | | | AR Endpoint |
| **NETWORK MANAGEMENT ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Role (PUSH PUBLISHER, PUSH SUBSCRIBER) |
| 2 | (m) | Attribute: | AREP State |
| 2 | (c) | Constraint | Role == PUSH SUBSCRIBER |
| 2.1 | (o) | Attribute: | Duplicate Pdu Detection Supported |
| 3 | (m) | Attribute: | DL Mapping Reference |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Unconfirmed Send |
| 2 | (o) | OpsService: | AR Compel |
| 3 | (o) | OpsService: | Get Buffered Message |
| 4 | (o) | OpsService: | Schedule Communication |
| 5 | (o) | OpsService: | Cancel Scheduled Sequence |

#### 6.3.7.3 Network management attributes

**Role**

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

PUSH-PUBLISHER Endpoints of this type publish their data issuing unconfirmed service Request-APDUs.

PUSH-SUBSCRIBER Endpoints of this type receive data published in confirmed service Response-APDUs.

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**DL Mapping Reference**

For PUSH-PUBLISHER AREPs, this attribute specifies the mapping to the transmit conveyance path. For PUSH-SUBSCRIBER AREPs, this attribute specifies the mapping to the receive conveyance path. DL mapping attributes for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

#### 6.3.7.4 Services

**Unconfirmed Send**

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

**AR Compel**

This optional service is used to cause an APDU locally buffered for transmission (for PUSH-PUBLISHER AREPs) or remotely buffered for transmission (for PUSH-SUBSCRIBER AREPs) in the data-link layer to be transmitted at the next available opportunity.

**Get Buffered Message**

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

**Schedule Communication**

This local service provides the AL user with the ability to schedule a sequence of DL-COMPELs for the AREP.

**Cancel Scheduled Sequence**

This local service provides the AL user with the ability to cancel an existing sequence which has previously been scheduled for the AREP.

### 6.3.8 Buffered network-scheduled bi-directional (BNB) AR endpoint class specification

#### 6.3.8.1 Class overview

This class is defined to support the cyclic exchange of confirmed services (only Read) and on-demand exchange of unconfirmed services between two application processes using buffers. The cycle is given by the network.

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

An AR ASE user wishing to convey a Read request or unconfirmed request APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU writes it into the data-link layer buffer, completely replacing the existing contents of the buffer. Furthermore, a copy of the APDU is stored at a local buffer within the AREP if the data-link layer buffer is current used by a precede request. The data-link layer transfers the buffer contents once at the earliest transfer by opportunity when a request to transmit (AR Compel) is received from the network.

At the receiving endpoint, the APDU is received from the network, it is written into the buffer, completely replacing the existing contents of the buffer. The endpoint notifies the user that the confirmed APDU (Read) has arrived and delivers it to the user according to the local user interface. Furthermore, a copy of the APDU is stored at a local buffer within the AREP. The following response from the AR ASE user is written by the AREP into the data-link layer buffer, completely replacing the existing contents of the buffer. The data-link layer transfers the buffer contents once at the earliest transfer opportunity when a request to transmit (AR Compel) is received from the network. The AR ASE user is notified again with the restored request when the response is sent over the network. This causes the AR ASE user to response again the current requested variable in a cyclic manner.

At the requesting endpoint, the APDU containing the response received from the network, it is written to the buffer, completely replacing the existing contents of the buffer. The endpoint notifies the user that the APDU has arrived and delivers it to the user according to the local user interface as the confirmation. If the AR ASE user wishes to convey a Read request again the response is generated locally from the data-link layer buffer. The content of the buffer is cyclic updated by the network with the related mechanism at the receiving endpoint.

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

Additionally the AREP monitors the connection by observing the network activities.

The following summarizes the characteristics of this AREP class.

Roles: Peer
Cardinality: 1-to-1

Timeliness: No

#### 6.3.8.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | | | Buffered Network-Triggered Bi-directional AR Endpoint |
| **CLASS ID:** | | | 44 |
| **PARENT CLASS:** | | | AR Endpoint |
| **NETWORK MANAGEMENT ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Role (CLIENT, SERVER, PEER) |
| 2 | (m) | Attribute: | AREP State |
| 3 | (m) | Attribute: | Remote Address Configuration Type (FREE,LINKED) |
| 4 | (m) | Attribute: | Initiator (TRUE/FALSE) |
| 5 | (m) | Attribute: | Receive DL Mapping Reference |
| 6 | (m) | Attribute: | Transmit DL Mapping Reference |
| 7 | (o) | Attribute: | Remote AP Name |
| 8 | (m) | Attribute: | CIN |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Unconfirmed Send |
| 2 | (o) | OpsService: | Confirmed Send |
| 3 | (o) | OpsService: | Establish |
| 4 | (o) | OpsService: | Abort |

#### 6.3.8.3 Network management attributes

**Role**

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

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

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Remote Address Configuration Type**

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

FREE The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the request service.

LINKED The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6-5).

**Initiator**

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR should be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

**Receive DL Mapping Reference**

This attribute provides a reference to the underlying data-link layer mapping for the conveyance path from the Publishing Manager. DL mapping attributes for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**Transmit DL Mapping Reference**

This conditional attribute provides a reference to the underlying data-link layer mapping for the transmit conveyance path. It is present only for PULL PUBLISHERs and PULL PUBLISHING MANAGERs. DL mapping attributes for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

**Remote AP Name**

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

**CIN**

This attribute (CIN – Control Interval Network triggered) indicates the control interval for monitoring a network triggered connection.

#### 6.3.8.4 Services

**Unconfirmed Send**

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

**Confirmed Send**

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

**Establish**

This service is used to establish an AR.

**DeEstablish**

This service is used to de-establish (gracefully terminate) an AR.

**Abort**

This service is used to terminate an AR.

### 6.3.9 Buffered network-scheduled and unscheduled uni-directional (BNU-Mp) AR endpoint class specification

#### 6.3.9.1 Class overview

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

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

The push model is used for the scheduled traffic, as follows.

An AR ASE user wishing to broadcast new data, uses an AR ASE Service Data Unit with a special encoding (see IEC 61158-6-5) to its AREP for distribution. The AREP sending the data writes it into the data-link layer buffer, completely replacing the existing contents of the buffer. The data-link layer transfers the buffer contents at the next scheduled opportunity. When the buffer contents are transmitted, the AR ASE notifies the user of the transmission.

If the AREP that sent data receives from the AR ASE user another data before the buffer contents are transmitted, the buffer contents will be replaced with the new data and the previous data will be lost.

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

The pull model is used for the on-demand traffic, as follows.

An AR ASE user wishing to send or read the remote value of the buffer, uses the Remote-Write or Remote-Read services with an AR ASE Service Data Unit specially encoded (see IEC 61158-6-5). An AR ASE user wishing to broadcast data on-demand writes it into the data-link layer buffer replacing the existing contents of the buffer. The data-link layer transfers the buffer contents after soliciting a transmission opportunity. The transmission of the buffer is reported to the user as the confirmation of the service. At the receiving endpoint, the data is received from the network and is immediately written into the buffer, completely overwriting the existing contents of the buffer. The endpoint notifies the user that the data has arrived and delivers it to the user. Extra local readings of the contents of the buffer are possible.

An AR ASE user wishing to read the remote value of a buffer, asks the data-link layer to solicit a transmission opportunity to get the value of the remote buffer. The transmission of the buffer is reported to the user as the confirmation of the service. At the receiving endpoint, the data is received from the network and is immediately written into the buffer, completely overwriting the existing contents of the buffer. The endpoint notifies the user that the data has arrived and delivers it to the user. Extra local readings of the contents of the buffer are possible.

This type of AR also supports, as an option, the ability of the AR ASE to determine publishing, transmission, or reception timeliness. For publishing timeliness, the sending AR ASE will mark the buffer if it has not been updated within a specified update interval. For reception timeliness, the receiving AR ASE will mark the buffer if it has not been received within a specified reception interval. Both publishing and reception timeliness information are given. For transmission timeliness, the sending AR ASE determines if the buffer has been transmitted within a specified interval. If the transmission does not occur, the FAL user is notified and the buffer contents are updated to indicate the lack of transmission timeliness.

Residence timeliness is supported by this type of AREP. Figure 7 illustrates this type of timeliness.

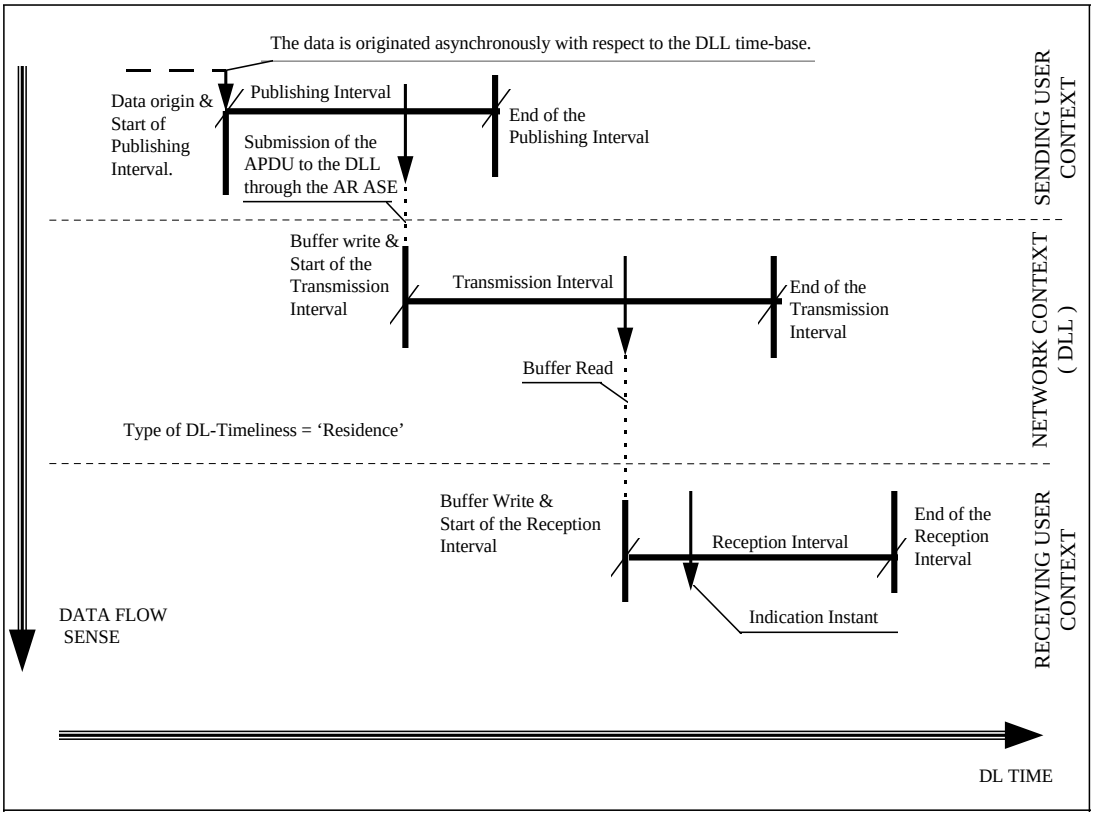



**Figure 7 – Residence timeliness**

This type of timeliness requires the use of a data link timer associated with data buffer (transmit or receive). The timer is started when the data is written into the transmit buffer by the publisher AREP or received from the network into the subscriber buffer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows.

Publisher AREP Data is timely if it is transmitted within the timeliness window after it is placed into the data link buffer for transmission. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate its current status.

Subscriber AREP Data is timely if it is received as timely by the data-link layer and is read from the receive buffer by the AREP within the timeliness window that was started when the data was received. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent reads from the buffer of the same data will indicate its current status.

Synchronized timeliness is supported by this type of AREP. Figure 8 illustrates this type of timeliness.

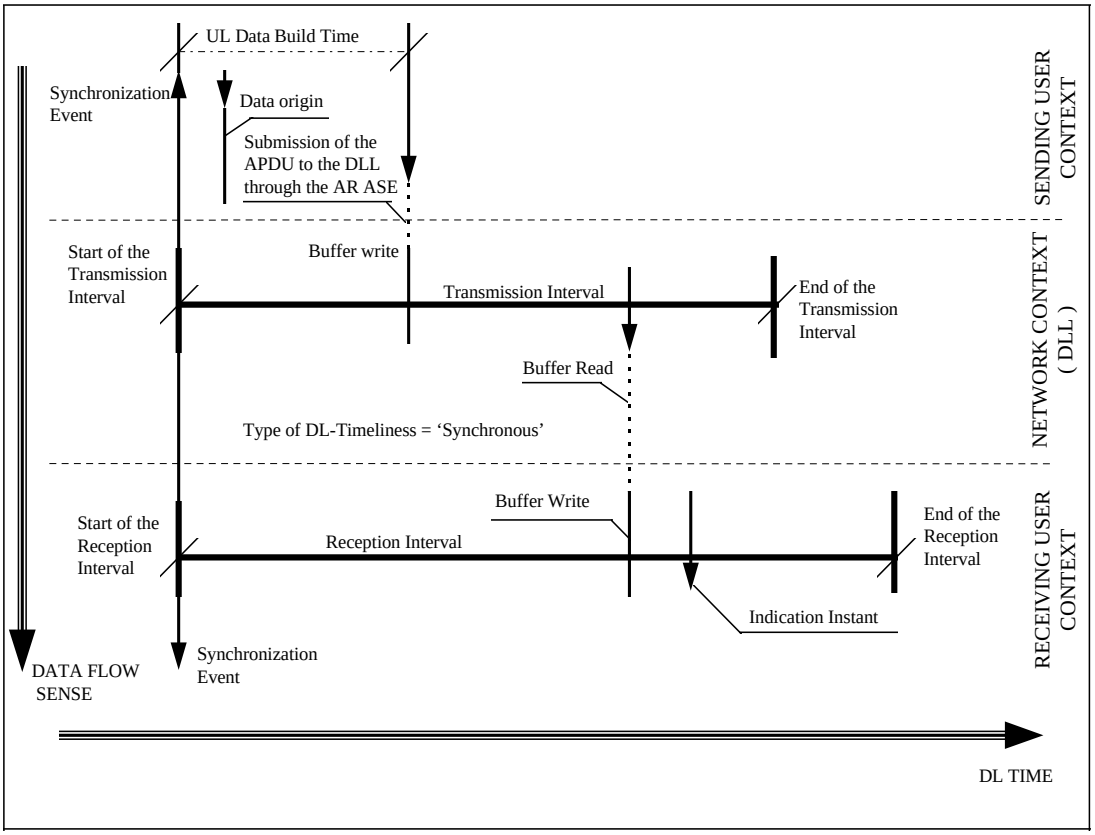



**Figure 8 – Synchronized timeliness**

This type of timeliness requires the use of two supporting data link connections. One is used to publish the data and the other is used to distribute a "sync mark" within the data-link layer to each of the stations participating as either a publisher or subscriber. The sync mark causes a timer associated with data buffer (transmit or receive) to be started within the data-link layer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows.

Publisher AREP data is timely if it is placed into the buffer within the time window after the receipt of a "sync mark". When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate the current status of the buffer.

Subscriber AREP Data is timely if it is received as timely by the data-link layer and is received within the time window. When the timeliness window expires, the data in the buffer is regarded as NOT Timely. Data received after the expiration of the window, but before the next sync mark, is also considered NOT Timely. Subsequent reads from the buffer of the same data will indicate its current status.

The following summarizes the characteristics of this AREP class.

Roles: Push Publisher
Push Subscriber

Cardinality: 1-to-Many

Timeliness: Optional

#### 6.3.9.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | | | Buffered Network-Scheduled Uni-directional AR Endpoint |
| **CLASS ID:** | | | 39 |
| **PARENT CLASS:** | | | AR Endpoint |
| **NETWORK MANAGEMENT ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | Role (PUSH PUBLISHER, PUSH SUBSCRIBER, PULL PUBLISHER, PULL SUBSCRIBER) |
| 2 | (m) | Attribute: | AREP State |
| 3 | (m) | Attribute: | DL Mapping Reference |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Unconfirmed Send |
| 2 | (o) | OpsService: | AR Compel |
| 3 | (o) | OpsService: | Get Buffered Message |
| 4 | (o) | OpsService: | Schedule Communication |
| 5 | (o) | OpsService: | Cancel Scheduled Sequence |
| 6 | (o) | OpsService: | Remote Read |
| 7 | (o) | OpsService: | Remote Write |

#### 6.3.9.3 Network management attributes

**Role**

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

PUSH-PUBLISHER Endpoints of this type publish their data issuing unconfirmed service Request-APDUs.

PUSH-SUBSCRIBER Endpoints of this type receive data published in confirmed service Response-APDUs.

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**DL Mapping Reference**

For PUSH-PUBLISHER AREPs, this attribute specifies the mapping to the transmit conveyance path. For PUSH-SUBSCRIBER AREPs, this attribute specifies the mapping to the receive conveyance path. DL mapping attributes for the data-link layer (IEC 61158-3-1) are specified in IEC 61158-6-5.

#### 6.3.9.4 Services

**Unconfirmed Send**

This optional service is used to send an unconfirmed service on an AR. The structure of the PDU is specified and described in the corresponding DL-Mapping State Machine.

**AR Compel**

This optional service is used to cause an APDU locally buffered for transmission (for PUSH-PUBLISHER AREPs) or remotely buffered for transmission (for PUSH-SUBSCRIBER AREPs) in the data-link layer to be transmitted at the next available opportunity.

**Get Buffered Message**

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

**Schedule Communication**

This local service provides the AL user with the ability to schedule a sequence of DL-COMPELs for the AREP.

**Cancel Scheduled Sequence**

This local service provides the AL user with the ability to cancel an existing sequence which has previously been scheduled for the AREP.

**Remote Read**

This service (based on the Pull Subscriber model) allows a user to read the contents of a remote buffer. The structure of the PDU is described and specified in the corresponding DL-Mapping State Machine in the protocol document.

**Remote Write**

This service (based on the Pull Subscriber model) allows a user to write the contents of a remote buffer. The structure of the PDU is specified and described in the corresponding DL-Mapping State Machine in the protocol document.

## 6.4 Summary of FAL classes

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

Table 80 provides a summary of the classes.

**Table 80 – FAL class summary**

| FAL ASE | Class | Class ID |
|---|---|---|
| Application Process | Application Process | 16 |
| | | |
| Data type | Fixed Length & String Data type | 5 |
| | Structure Data type | 6 |
| | Array Data type | 12 |
| | | |
| Application Relationship | AREP | 32 |
| | QUB-Co | 34 |
| | QUU | 36 |
| | BNU | 38 |
| | BNU Mp | 39 |
| | BUU (for future study) | 40 |
| | BUB | 42 |
| | BNB | 44 |
| | QUB-Cl | 45 |
| | QUB-FC | 46 |
| | QUB-Seg | 48 |
| | | |

| FAL ASE | Class | Class ID |
|---|---|---|
| Variable | Fixed Length & String Variable | 7 |
| | Array Variable | 8 |
| | Data Structure Variable | 9 |
| | Variable List | 10 |
| | | |
| Event | Event | 4 |
| | Event List | 17 |
| | Notifier | 18 |
| | | |
| Load Region | Load Region | 2 |
| | | |
| Function Invocation | Function Invocation | 3 |
| | Action | 19 |

## 6.5 Permitted FAL services by AREP role

Table 81 below defines the valid combinations of services and AREP roles (which service APDUs and AREP with the specified role can send or receive). The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

**Table 81 – Services by AREP role**

| | Unc | Cnf | Client | | Server | | Push Publ | | Push Subsc | | Pull Publ Mgr | | Pull Publ | | Pull Subsc | | Report Src | | Report Sink | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **FAL Services** | | | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** |
| **Mgt ASE** | | | | | | | | | | | | | | | | | | | | |
| Create | | X | X | | X | | | | | | | | | | | | | | | |
| Delete | | X | X | | X | | | | | | | | | | | | | | | |
| Get Atributes | | X | X | | X | | | | | | | | | | | | | | | |
| Get Attribute List | | X | X | | X | | | | | | | | | | | | | | | |
| Set Attributes | | X | X | | X | | | | | | | | | | | | | | | |
| Begin Set Attributes | | X | X | | X | | | | | | | | | | | | | | | |
| End Set Attributes | | X | X | | X | | | | | | | | | | | | | | | |
| **AP ASE** | | | | | | | | | | | | | | | | | | | | |
| Subscribe | | X | X | | X | | | | | | | | | | | | | | | |
| Identify | | X | X | | X | | | | | | | | | | | | | | | |
| Get Status | | X | X | | X | | | | | | | | | | | | | | | |
| Status Notification | X | | | | | | X | | X | | | | | | | | X | | X | |
| Initiate | | X | X | | X | | | | | | | | | | | | | | | |
| Terminate | | X | X | | X | | | | | | | | | | | | | | | |
| Conclude | | X | X | | X | | | | | | | | | | | | | | | |
| Reject | | X | | X | | | | | | | | | | | | | | | | |
| **AR ASE** | | | | | | | | | | | | | | | | | | | | |
| AR-Confirmed Send | | | X | | X | | | | | | | | | | | | | | | |
| AR-Unconfirmed Send | | | X | X | X | X | X | | X | | | | | | | | | | | |
| AR-Establish | | | X | | X | | | | | | | | | | | | | | | |
| AR-De-Establish | | | 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-Schedule Communication | | | | | | | X | | X | | | | | | | | | | | |
| AR-Cancel Schedule | | | | | | | X | | X | | | | | | | | | | | |

| | Unc | Cnf | Client | | Server | | Push Publ | | Push Subsc | | Pull Publ Mgr | | Pull Publ | | Pull Subsc | | Report Src | | Report Sink | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **FAL Services** | | | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** |
| Sequence | | | | | | | | | | | | | | | | | | | | |
| AR-Status | | | X | | X | | | | | | | | | | | | | | | |
| AR-XON-OFF | | | | X | X | | | | | | | | | | | | | | | |
| AR Remote Read | | | | | | | | | | | X | | X | | X | | | | | |
| AR Remote Write | | | | | | | | | | | X | | X | | X | | | | | |
| Variable ASE | | | | | | | | | | | | | | | | | | | | |
| Read | | X | X | | X | | | | | | X | X | | X | | | | | | |
| Read List | | X | X | | X | | | | | | | | | | | | | | | |
| Write | | X | X | | X | | | | | | | | | | | | | | | |
| Write List | | X | X | | X | | | | | | | | | | | | | | | |
| Information Report | X | | | | | | X | | X | | | | | | | | X | | X | |
| Information Report List | X | | | | | | X | | X | | | | | | | | X | | X | |
| Exchange | | X | X | | X | | | | | | | | | | | | | | | |
| ExchangeList | | X | X | | X | | | | | | | | | | | | | | | |
| Event ASE | | | | | | | | | | | | | | | | | | | | |
| Unconfirmed Ack Events | X | | | | | | X | | X | | | | | | | | X | | X | |
| Confirmed Ack Event | | X | X | | X | | | | | | | | | | | | | | | |
| Ack Event List | | X | X | | X | | | | | | | | | | | | | | | |
| Enable Event | | X | X | | X | | | | | | | | | | | | | | | |
| Enable Event List | | X | X | | X | | | | | | | | | | | | | | | |
| Get Event Summary | | X | X | | X | | | | | | | | | | | | | | | |
| Get Event Summary List | | X | X | | X | | | | | | | | | | | | | | | |
| Query Event Summary List | | X | X | | X | | | | | | | | | | | | | | | |
| Event Notification | X | | | | | | X | | X | | | | | | | | X | | X | |
| Notification Recovery | X | | | | | | X | | X | | | | | | | | X | | X | |
| Load Region ASE | | | | | | | | | | | | | | | | | | | | |
| Initiate Load | | X | X | | X | | | | | | | | | | | | | | | |
| Push Segment | | X | X | | X | | | | | | | | | | | | | | | |
| Pull Segment | | X | X | | X | | | | | | | | | | | | | | | |
| Terminate Load | | X | X | | X | | | | | | | | | | | | | | | |
| Discard | | X | X | | X | | | | | | | | | | | | | | | |
| Function Invocation ASE | | | | | | | | | | | | | | | | | | | | |
| Start | | X | X | | X | | | | | | | | | | | | | | | |
| Stop | | X | X | | X | | | | | | | | | | | | | | | |
| Resume | | X | X | | X | | | | | | | | | | | | | | | |
| Reset | | X | X | | X | | | | | | | | | | | | | | | |
| Kill | | X | X | | X | | | | | | | | | | | | | | | |
| Action Invoke | X | | | | | | | | | | | | | | | | X | | X | |
| Action Return | X | | | | | | | | | | | | | | | | X | | X | |

# 7 Type 5 communication model specification

## 7.1 Concepts

### 7.1.1 Overview

This subclause introduces the concept of the FAL application entity (FAL AE). It is referred to as the Field Device Access (FDA) Agent. The FDA Agent is composed of Type 5 specific APO ASEs and ARs and a subset of the Type 9 APO ASEs. A subset of the data types defined in Clause 4 are supported. The Type 9 ASEs, classes, and services supported by Type 5 are defined in 14.3.

The FDA Agent maps the Type 5 and Type 9 ASEs onto a set of underlying connection-oriented and connectionless communication service endpoints modelled as sockets. These services are referred to as socket services in the remainder of this clause.

Socket services are independent of the actual protocols used to provide their services. Any protocols that can be mapped to these services can be used. Sockets are addressed by network address and port number.

Type 5 supports application access to both Type 9 and Type 5 field devices. Access to Type 9 field devices is provided through one class of Type 5 devices called a *Linking Device.* Linking Devices contain a FDA Agent that provides a gateway function between Type 5 and Type 9 networks.

Another class of Type 5 devices is capable of interconnecting a Type 5 network to links that support other data link layer types. These Type 5 devices are called Gateway Devices. The FDA Agent in Gateway Devices provides gateway services that map Type 5 APDUs to their counterparts on the other links.

Another class of Type 5 devices is capable of supporting redundant LAN connections. These Type 5 devices are called LAN Redundancy Capable Devices. The FDA Agent in these devices provides LAN Redundancy ASE services to the LAN Redundancy entity in the device.

The FDA Agent uses a Type 5 MIB to store data. The specification of this MIB is beyond the scope of this standard.

Throughout the remainder of this clause, Type 5 devices are FAL devices that contain a FDA Agent and Type 9 Devices refer to devices that implement the Type 9 FAL and the Type 1 DLL.

### 7.1.2 Objectives of the FDA agent

The primary objective of the FDA Agent is to convey Type 5 and Type 9 ASE services using socket services. This allows Type 5 and Type 9 field devices, conventional I/O devices, and other I/O devices to be connected through a FDA Agent in a *Linking* or *Gateway Device*.

Another objective of the FDA Agent is to republish Type 9 data from one Type 9 link onto another in multi-ported Type 9 devices that do not support Type 9 data link layer bridging. This allows a FDA Agent in a Linking Device to integrate multiple standalone Type 9 interfaces instead of using a Type 9 bridge.

Another objective of the FDA Agent is to send and receive messages to support applications responsible for adding and deleting Type 5 application objects to/from the network, and for locating application objects on the network.

Another objective of the FDA Agent is to send and receive diagnostic messages to support LAN redundancy applications.

As a result, the FDA Agent enables:

- construction of control systems from Type 5 and Type 9 devices,
- linking of Type 5 and Type 9 networks by Linking Devices, and
- accessing by remote applications of field devices of any type through socket services using the Type 5 Communications Model.

### 7.1.3 Data type ASE

#### 7.1.3.1 Data types

The data types supported are a subset of those defined in Clause 4. They are:

- Boolean
- Integer8
- Integer16
- Integer32
- Unsigned8
- Unsigned16
- Unsigned32
- Floating Point
- VisibleString
- OctetString
- Date
- TimeOfDay
- TimeDifference
- BitString
- TimeValue

#### 7.1.3.2 Nesting level

One level of nesting is supported. That is, constructed types may not contain constructed types.

### 7.1.4 APO ASEs

#### 7.1.4.1 Overview

The FDA Agent supports Type 5 and Type 9 APO ASEs as specified in 14.3. The Type 5 ASEs are the VFD, SMK and LAN Redundancy ASEs. The Type 5 VFD ASE is an adaptation of the Type 9 VFD ASE. The FDA Agent also supports versions of the QUB-CO and QUU Type 9 ARs that are adapted for use over sockets.

#### 7.1.4.2 LAN redundancy ASE

The LAN Redundancy ASE specifies services that support LAN Redundancy entities. A LAN Redundancy Entity in LAN redundancy capable devices uses the services of the FDA Agent to send and receive LAN redundancy ASE APDUs. These APDUs are used by the LAN Redundancy Entity to construct a network status table used to choose routes that avoid network faults.

#### 7.1.4.3 System management kernel (SMK) ASE

##### 7.1.4.3.1 SMK

The SMK entity is the entity in a device that is responsible for system management operation in a device. The SMK ASE defines a set of services that support the conveyance of system management requests and responses to and from SMK entities.

When resident in a Type 5 device, the SMK services are conveyed by Type 5 ARs. When resident in a device of another type, the SMK services may be conveyed by direct access to the data link layer connectionless services.

Like Type 9 ASE services, SMK ASE services can be conveyed between a device connected to a Type 5 network and a device connected to a network that uses an underlying Type 9 data-link layer. In this case, the FDA Agent in a Linking Device receives an SMK service request or response from one network and maps it for conveyance by the other.

#### 7.1.4.3.2 SMK ASE services

One SMK ASE service is defined for identifying devices.

Two SMK ASE services are defined for adding and deleting Type 5 Devices and Type 9 devices connected to a Linking Device.

– One of these services is used by the SMK in a new device to announce its presence periodically on the network.
– The other service is used by a configuration application receiving the annunciation to return basic configuration information to the SMK. This allows the SMK to learn its role in the system and begin normal operations.

Two more SMK ASE services are defined for finding devices and named objects on the network.

### 7.1.4.4 Virtual field device (VFD) ASE

#### 7.1.4.4.1 The VFD object

The Type 9 VFD Object is used in Type 5. In Type 5, all objects and services defined by the Type 9 ASEs are used to access APOs within a VFD, with the following exceptions. The Type 5 VCR replaces the Type 9 VCR and the Type 5 Initiate service replaces the Type 9 Initiate service. All Type 5 and Type 9 VFD services are conveyed on Type 5 networks using the Type 5 syntax and procedures defined in IEC 61158-6-5.

#### 7.1.4.4.2 Virtual communication relationships (VCRs)

##### 7.1.4.4.2.1 Overview

Type 5 VCRs are similar to the Type 9 VCR (defined by the Type 9 Context Management ASE). Type 5 VCRs provide a Type 9 service interface for access to both Type 5 VFDs and Type 9 VFDs. Type 9 VFDs are accessed through Type 5 AEs in Linking Devices. Three types of VCRs are defined, Client/Server, Publisher/Subscriber, and Report Distribution.

##### 7.1.4.4.2.2 Client/server VCRs

The FDA Agent supports dynamically established client/server VCRs, as follows. Clients send requests to initiate VCRs to the server FAL AE. The server FAL AE dynamically creates a new server VCR and delivers the request to it for processing.

##### 7.1.4.4.2.3 Publisher/subscriber VCRs

Each subscriber VCR endpoint is configured to receive from a specific subscriber AR. Each subscriber VCR endpoint is configured to receive its messages from a single publisher VCR endpoint. The original publisher may exist on a Type 9 link when a Linking Device is used to republish the data onto the Type 5 network. In addition, if the subscriber is located in a Linking Device, it may be configured to republish the data onto a Type 9 link.

When conveyed through a Linking Device, the FAL AE maps Type 9 publisher/subscriber VCRs to Type 5 publisher/subscriber VCRs, receiving published data on one and republishing it on the other.

The FDA Agent can be configured for three types of publishing:

– A local Type 5 application can publish data onto the Type 5 network through the FDA Agent.
– The FDA Agent in a Linking Device can republish data from a Type 9 device onto the Type 5 network.
– The FDA Agent in a Linking Device can republish data from a Type 5 device onto a Type 9 link.

#### 7.1.4.4.2.4 Report distribution VCRs

Report distribution VCRs operate analogously to publisher/subscriber VCRs.

The FDA Agent can be configured for three types of report distribution:

– a local Type 5 application can send reports onto the Type 5 network through the FDA Agent;
– the FDA Agent in a Linking Device forward reports received from devices on a Type 9 link onto the Type 5 network;
– the FDA Agent in a Linking Device can forward reports received from a Type 5 device onto a Type 9 link.

#### 7.1.4.4.3 Initiate service

The Initiate service can be directed to a VFD's generic address. This allows multiple client applications to open concurrent access to the VFD using the same address. The VFD ASE operating at that address creates and a new VCR dynamically to process each received initiate request.

The Initiate service also can be directed to the Management Information Base (MIB) VFD of a device without knowing its generic address. Only the device address is necessary to open the AR to the device.

The Initiate service operates locally for publisher/subscriber VCRs. It opens the VCR for publishing or subscribing.

### 7.1.5 Application relationships

#### 7.1.5.1 Sessionless transfers

The FDA Agent sends and receives SM and LAN Redundancy messages without FDA sessions using underlying connectionless services. Sessionless transfers are not capable of concatenating messages.

#### 7.1.5.2 FDA sessions and Type 5 VCRs

Type 5 VCRs provide access to VFDs across the Type 5. They are the Type 5 counterpart to Type 9 VCRs. Type 5 VCRs use FDA Sessions to transfer Type 9 services.

FDA sessions provide communications to/from FDA Agents. They are the Type 5 counterpart to Type 9 ARs. They are established for the life of the session to operate over either a TCP connection or over connectionless services, between source and destination ports.

The similarities between Type 9 ARs and FDA Sessions are as follows.

– Type 9 ARs and FDA Sessions support the same VCR models, client/server, publisher/subscriber, and Report Distribution.

– Type 9 devices and FDA Agents each may contain one or more Type 9 AR/FDA Session endpoints.

– Type 9 ARs are identified on the wire by source and destination DLCEPs/DLSAPs. FDA Sessions are identified on the wire by source and destination network addresses (network address plus port number).

– Type 9 SMKs communicate directly with the data link layer using the Unitdata service. FDA SM messages are conveyed in a similar fashion, using connectionless services without using a session.

NOTE Type 9 ARs and FDA sessions are negotiated to operate using a specific version of their protocol data units (PDUs)/messages. All PDUs/ messages sent on a Type 9 AR/ session use the same protocol version. FDA Agents that support newer versions also support previous versions for backward compatibility.

– Type 9 ARs and FDA Sessions differ in the following ways.

– A single session is capable of supporting more than one Type 5 VCR. Type 9 ARs are capable of supporting only a single Type 9 VCR.

– Sessions are capable of concatenating multiple messages together into a single PDU. Each DLL PDU conveys only a single Type 9 message (PDU).

– Type 5 connectionless client/server sessions are established using underlying connectionless services. For Type 9 ARs, there is no concept of a connectionless client/server AR.

– Connection-oriented client/server sessions are established over TCP. Once the TCP connection has been established, the client sends an FDA Open Session request message on that connection to open the session. Type 9 ARs operate similarly, but the connection is established at the data link layer, and the data link connect request contains the FAS request to open the AR.

#### 7.1.5.2.1 Client/server

**Client/server sessions**

Client/server sessions are used to support client/server VCRs. A single client/server session may support more than one VCR. Client/server sessions should be established before they can be used. They are established by exchanging FDA Open Session request and Response messages between client and server.

If TCP is to be used, the client first sends a TCP connect request to the reserved FDA Port of the server device. The FDA Agent at the server listens at the reserved FDA Port for TCP connect requests, and when one is received it creates a new server session endpoint. It then waits for the next TCP connect request, while the newly created session endpoint waits for an Open Session request.

If underlying connectionless services are to be used, the client sends the Open Session request message to the reserved Type 9 port of the server device. The FDA Agent at the server listens at the reserved FDA Port for Open Session requests, and when one is received it creates a new server session endpoint, allocates a connectionless socket for it with a new port address, and delivers the Open Session request to it. If creation of a new server session endpoint is not possible, a negative response is sent from the FDA Port. The FDA Agent then waits for the next Open Session request.

When the new server session endpoints receives the Open Session request it processes the message. The FDA Open Session request message contains negotiable parameters that allow the server to dynamically configure the server session endpoint and that indicate whether or not the session will be used as a *configuration* session (see 7.1.5.4).

If the request is accepted, the newly created session endpoint returns a positive response. If the request is rejected, the newly created session endpoint returns a negative response, closes the underlying socket, and deletes itself. Responses are returned from the underlying socket, or from the reserved Type 9 port if it was not possible to open a new connectionless socket.

Client/server sessions are terminated by user request, lack of activity, or because of FDA protocol problems.

**Client/server VCRs**

Client/server VCRs are used to convey requests and responses to and from VFDs. Client/server VCRs may be established between a client application and either a Type 5 or a Type 9 VFD. Type 9 VFDs are accessed by establishing a Type 5 VCR to a linking device. This Type 5 VCR endpoint accesses a Type 9 VCR that connects to the Type 9 VFD. Multiple Type 5 VCRs, over the same or different sessions, may concurrently access a single Type 9 VCR. Type 9 field devices, therefore, do not have to provide individual server VCR endpoints to support each concurrent Type 5 client.

Client/server VCRs should be established before they can be used. Client/server VCRs are established by exchanging Initiate request and response messages between client and server after a session between them has been opened.

The client sends the Initiate request message to the *generic* VCR endpoint for the server VFD. The generic VCR endpoint is identified in the Initiate by a *generic* selector or by a special value that indicates *NMA Access*. The generic selector value is contained in the VFD List in the MIB for the device. It can also be obtained using the Find Tag Query for the tag of the VFD or for an object contained within the VFD.

The generic VCR endpoint listens at this selector for Initiate requests, and when one is received it creates a new server VCR endpoint and adds an entry in MIB list of *automatic* VCRs. The generic VCR endpoint also delivers the Initiate request to the new VCR endpoint, and then waits for the next Initiate request.

The new VCR endpoint uses address information contained in the request to determine whether the VFD is a local Type 5 VFD, or in the case of a linking device, whether it is a Type 9 VFD. If it is a local Type 5 VFD or if it is a Type 9 management agent VFD in the linking device, then the VCR endpoint delivers an Initiate indication primitive and waits for the response. If the VFD is a Type 9 VFD located in a Type 9 device connected to the linking device, the VCR endpoint submits an Initiate request primitive to the client Type 9 VCR endpoint if the client Type 9 VCR endpoint is not already open. If the client Type 9 VCR endpoint is open, the local Type 5 VCR endpoint locally attaches itself to it.

If the request is accepted by the local VFD or if the Type 9 VCR is opened or already open, the newly created VCR endpoint returns a positive response that contains the OD Index for its new entry in the MIB list of automatic VCRs. This value is used to identify the VCR endpoint in all future messages sent on the VCR. If the request was sent using NMA Access and the underlying session is not a configuration session, then the VCR endpoint rejects all update requests that it receives from the client.

If the Initiate request is rejected, the newly created VCR endpoint returns a negative response, removes its entry in the MIB list of automatic VCRs, and deletes itself.

Client/server VCRs are terminated in the same ways that Type 9 VCRs are terminated. In addition, they are terminated when there is no longer activity on them. The states and processing associated with the VCR initiation and termination are described in 61158-6-5.

Figure 9 illustrates the use of the session to open Type 5 and Type 9 VCRs.

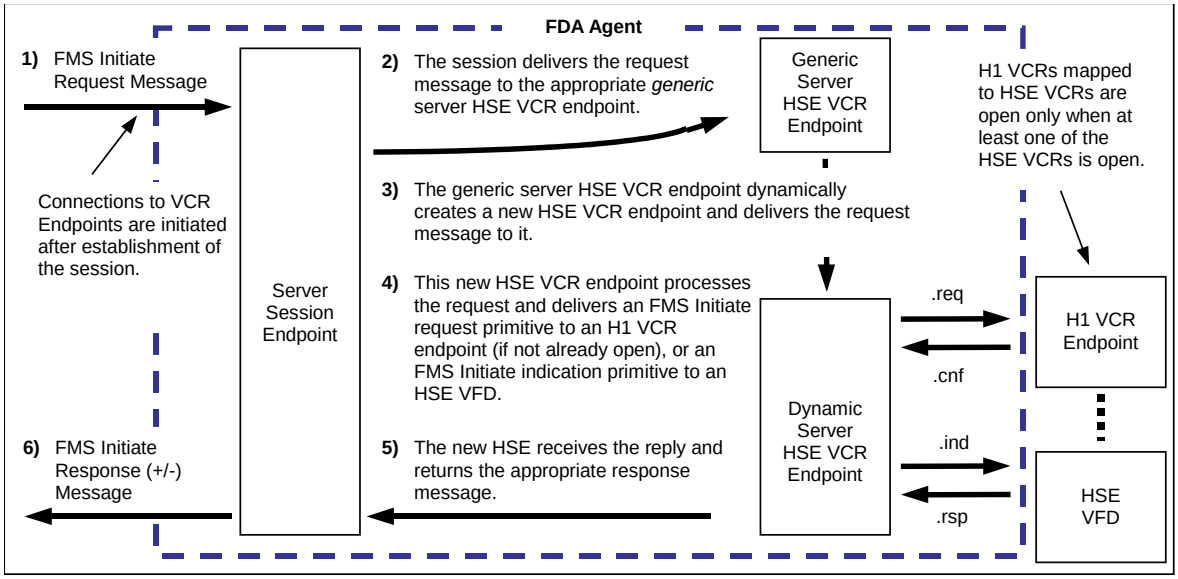



**Figure 9 – VCR initiation**

#### 7.1.5.2.1.1 Keeping client/server VCRs and their sessions open

Client/server sessions and VCRs close automatically when there is no message traffic for a period of time known as the Inactivity Close Time. To prevent VCRs from terminating due to inactivity, the client VCR endpoint sends a confirmed Idle request message when its Idle timer expires. Its Idle timer is restarted using no more than of one third of its inactivity close time each time it sends a message. The server VCR endpoint receiving the Idle request message returns the corresponding Idle Response message immediately. When there are no active VCRs on a session, no Idle messages are sent. This causes the session to timeout and close.

### 7.1.5.2.2 Publisher/subscriber

#### 7.1.5.2.2.1 General

Publisher/subscriber sessions have been adapted from their counterpart Type 9 ARs. Publisher/subscriber sessions operate as follows.

- Messages are not sent by publisher or subscriber endpoints to establish sessions, in contrast to Type 9 publisher/subscriber connections.
- Published data is multicast using underlying connectionless services to a configured multicast address and to a configured port number.
- Publishers use unconfirmed service messages only to publish data.
- Each Device may be configured with zero, one, or more publisher session endpoints used to transmit published data.
- Subscribers are unknown to the publisher.
- There may be 0, 1, or more subscribers receiving published data.
- Each Device may be configured with zero, one, or more subscriber session endpoints used to receive published data.

Publisher/subscriber VCRs have been adapted from their Type 9 counterpart VCRs. Publisher/subscriber VCRs operate as follows.

- Messages are not sent by publisher or subscriber endpoints to establish VCRs.
- Detection of lost messages, duplicate messages, and late messages is possible, although recovery of lost messages is not possible. However, applications may have their own methods for recovering lost information (e.g. by accessing trend reports).

- Zero, one, or more publisher VCR endpoints may use the same publisher session endpoint to publish their data.
- A publisher VCR endpoint may be configured to publish data for a FBAP.
- A publisher VCR endpoint may be configured to republish data received by a Type 9 subscriber VCR endpoint.
- Each subscriber VCR endpoint receives data from a single publisher VCR endpoint.
- A subscriber VCR endpoint may be configured to deliver received data to a local application. A separate subscriber VCR endpoint is required for each local application that is to receive the published data.
- A subscriber VCR endpoint may be configured to deliver received data to a Type 9 publisher VCR endpoint, causing it to republish Type 5 data onto a Type 9 link. A separate subscriber VCR endpoint is required for each Type 9 link that is to republish the data.

#### 7.1.5.2.2.2 Session and VCR establishment

Publisher/subscriber endpoints are established locally. There are no FDA messages exchanged to establish them.

#### 7.1.5.2.2.3 Timeliness

Applications submit published data to the VCR configured in their link objects. The VCR submits it to its publisher session. The publisher session buffers it for up to the Transmit Delay Time defined for the session prior to submission to Subscriber endpoints deliver the received data to the subscribing applications or to Type 9 for republication (in linking devices) immediately upon receipt. Timeliness of published data is therefore dependent on the publisher Transmit Delay Time, on the latency of the network, and on the delay imposed by the underlying connectionless services and the FDA Agent within the receiving device.

The FDA Message Trailer also contains an optional timestamp that can be delivered to subscribing applications to allow them to determine whether or not the data was published within an acceptable time-window. This timestamp is applied to the message by the session endpoint at the time it submits the message for transfer. Use of this timestamp is configurable.

The timestamp is passed between the FDA Agent and the publishing/subscribing application through local means (they are not part of the Service interface).

#### 7.1.5.2.3 Report distribution

Report Distribution sessions are identical to publisher/subscriber sessions with the following exceptions.

- Report sources correspond to publishers.
- Report sinks correspond to subscribers.
- Reports are identified by the FDA Address in the FDA Message header and by the *index* parameter in the FDA Portion of the message.
- Each report sink session endpoint delivers all received messages to all VCRs associated with it.

#### 7.1.5.2.4 Terminating sessions

Sessions are terminated as a result of:

- Client/server sessions are terminated as a result of inactivity. Inactivity is defined as the failure of a session in the Open state to receive a valid message within the time period specified by the Inactivity Close Time session attribute.

- Client/server sessions are terminated as a result of termination of the underlying TCP connection.
- Client/server sessions are terminated as a result of receipt of an invalid message or a second Open Session request message. These messages are an indication that the data stream has lost synchronization or is communicating with a non-conformant partner. Whether or not subscriber session endpoints close as a result of receipt of an invalid message is implementation dependent.
- User request. User requests to close a session are implementation dependent.

When terminating a session because of inactivity, because of user request, or because of receipt of an invalid message or a second Open Session request message, the FDA Agent performs the steps below, in order. When terminating a session because the underlying TCP connection has been terminated, the FDA Agent performs steps 1, 4, and 5 only.

1) Stops accepting requests to send messages on the session,
2) Sends Type 9 Abort messages on all of its open Type 5 VCRs,
3) Terminates supporting TCP connections, if applicable.
4) Delivers all queued confirmation and error primitives, and discards all queued indication primitives,
5) Closes the corresponding VCR endpoints and issues a Type 9 Abort indication primitive for each.

NOTE Applications may close VCRs using the Type 9 Abort Service without closing the supporting FDA session.

#### 7.1.5.3 FDA agent initialization

When the FDA Agent starts up, it initializes its server session endpoints using default attribute values. Once initialized, it is ready to send and receive messages for the SMK and for the LAN Redundancy entity. Once SMK has reached its operational state, it enables the FDA Agent to send and receive messages for all other device applications/VFDs.

#### 7.1.5.4 Configuring sessions and VCRs

Before the device can become completely operational, its MIB needs to be configured. FDA session and VCR configuration information is contained in the MIB.

MIB configuration is performed using client/server *configuration* sessions.

Only NMA configuration sessions are allowed to change information in the MIB, in linking device Type 9 interface MIBs, and in Type 9 MIBs of Type 9 devices accessed through the linking device. Only one configuration session to a Type 5 device is allowed to be open at a time. After the configuration session has been opened, separate VCRs are opened to the Type 5 management agent and to each Type 9 management agent using the Initiate service using the NMA Access connect option.

Configuration sessions also provide configuration applications access to VFDs other than NMA VFDs. However, unlike access to the NMA VFDs, other sessions may concurrently write to these non-NMA VFDs.

Non-configuration client/server sessions provide read-only access to NMA VFDs after connecting to them using the Initiate service with the NMA Access connect option. Non-configuration client/server sessions provide read-write access to all other VFDs. The VCR is responsible for filtering update services submitted for MIB updating using these sessions. The following is the list of the update services that are filtered on these VCRs:

- Generic Initiate Download Sequence
- Generic Download Segment

– Generic Terminate Download Sequence
– Initiate Download Sequence
– Download Segment
– Terminate Download
– Request Domain Download
– Write
– Write with Subindex

NOTE Client/server sessions and client/server VCRs are defined by default and are not configured, unless they are used to support access from a Type 9 client across the Type 5 network. In this case, only the client endpoints are configured; the server endpoint acquires its attributes dynamically during endpoint establishment, as it does in all other cases.

Publisher/subscriber and Report Distribution Sessions and their associated VCRs are always configured.

#### 7.1.5.5 Conveying services

##### 7.1.5.5.1 General

The FDA Agent conveys FDA, VFD, SM and LAN Redundancy services and Type 9 services. VFD and Type 9 services are used to access VFDs and their objects. SM services are used to access SMKs. LAN Redundancy services are used to support redundant network interfaces. To convey these services the FDA Agent:

– receives service primitives from local applications and/or the Type 9 stack and converts them to FDA messages that it submits to the underlying socket interface.
– receives messages from its underlying socket interface and converts them into service primitives that it delivers to local applications and/or to the Type 9 stack. If a non-configuration session receives an update request message for the MIB, then the receiving VCR endpoint sends a negative response using error class = "access" and error code = "object access denied", instead of delivering the message to the MIB.
– receives SM service requests and responses that it submits to the underlying connectionless services interface.
– receives SM messages from its underlying connectionless services interface and delivers them as SM service indications and confirmations.
– receives LAN Redundancy service requests that it submits to the underlying connectionless services interface.
– receives LAN Redundancy messages from its underlying connectionless services interface and delivers them LAN Redundancyservice indications and confirmations.

The following subclauses discuss these topics in more detail.

##### 7.1.5.5.2 Buffered transfers

Session endpoints buffer data in accordance with their Transmit Delay Time attribute. This attribute allows endpoints to concatenate messages into a buffer and transfer them in a single transmission. See 7.1.5.7.4, Concatenating Messages, for a description.

##### 7.1.5.5.3 Matching requests and responses

As stated in 1.2, this standard does not address the method by which an associated protocol pairs a response with an inducing request. Thus the following description is strictly hypothetical.

When the FDA Agent receives a confirmed request message from its underlying socket, it delivers or forwards it to the destination VFD and waits for the response. When the response is received, it forwards the response to the requester.

A request instance ID may be carried in the Message Trailer to match the response received from the VFD with the corresponding request. A value is placed into the instance ID field of the request PDU and the same value is returned in this field in the response PDU. While a request is outstanding (response not yet received), the AE should not use the same Instance ID for another request that it sends to the same server using the same protocol identified in the Message Header (e.g. Type 9 or SM). That ID can be used, however, in requests sent to a different server, or to the same server using a different protocol.

The requesting AE determines each Instance ID value. The scope of the Instance ID is specific to the message type, as shown in Table 82.

**Table 82 – Scope of Invoke Id**

| Message Type | Instance ID is unique within the scope of: |
|---|---|
| Confirmed SM messages | the requesting socket |
| FDA Open Session and Initiate messages | the requesting socket |
| Other Confirmed messages | the associated VCR identified in the FDA Address field of the message header |

When a request is received by a linking device that it is to forward to an attached Type 9 link, it solves the following two problems.

First, to ensure uniqueness of the Instance ID on a Type 9 VCR, the FDA Agent maps Instance IDs received from different clients onto unique Type 9 VCR Instance IDs. Once this has been done, responses received on a Type 9 VCR are mapped back to the requesting client, and the originating instance ID is recovered. How the FDA Agent implements this mapping is not specified by this document.

Second, it aborts the Type 9 client/server VCR if the response is not returned within a time-out period specified in the MIB. After aborting the Type 9 VCR, it returns a response to the requester using error class = "service" and error code = "response time-out". It then attempts to reestablish the Type 9 VCR. If it is unsuccessful, it aborts all VCRs that are associated with the Type 9 VCR.

#### 7.1.5.5.4 Misordered messages

Misordered messages can be received when underlying connectionless services are used. Misordering means that consecutively received messages are not received in the same order that they were transmitted.

NOTE VCRs use message numbers to detect misordering. Message numbers are used to sequentially number each message transmitted by a VCR. Misordering is not detected for SM or LAN Redundancy message transfers.

Message numbers are an optional field defined for the Message Trailer. Their presence in the trailer is configured for publisher/subscriber and report distribution sessions and negotiated for client/server sessions. When they are defined for the session, misordering detection is enabled, regardless of the session endpoint type.

Using the message number, the types of misordering which can be detected are given in Table 83.

**Table 83 – Types of misordering detectable by message numbers**

| Misordering Type | Description |
|---|---|
| none | the next in-sequence message number is received |
| duplicate messages | the same message number received more than once |
| late messages | lower message number received after a higher message number |
| missed messages | a reasonable (see Note) gap in message numbers between consecutively received messages |
| loss of sync | an unreasonable gap in message numbers between consecutively received messages |

NOTE The dividing line between reasonable and unreasonable is termed the "guard band". A gap in received messages numbers that exceeds the guard band indicates that it is unreasonable to believe that the gap represents missed messages. Instead, it is an indicator that synchronization has been lost with the sender. This could occur because the sender has gone through a reset, or that the sender has failed and a backup sender has taken over. The guard band is discussed below.

All types of misordered messages received on client/server sessions or on report distribution sessions are delivered by the FDA Agent as they are received. This is the same as the delivery order provided when message numbers are not used.

For publisher/subscriber VCRs, the delivery of misordered message types is handled differently, as described in Table 84.

**Table 84 – Delivery of misordered message types on publisher/subscriber VCRs**

| Misordering Type | Receiving subscriber VCR endpoint processing |
|---|---|
| none | The in-sequence message is delivered |
| duplicate messages | The duplicate message is discarded |
| late messages | The late message is discarded |
| missed messages | The gap is ignored and the received message is delivered |
| loss of sync | The loss of sync is ignored and the received message is delivered |

For all types of VCRs, statistics can be gathered to support network management, as described in Table 85.

**Table 85 – Statistics gathered per VCR**

| Misordering Type | Statistic gathered |
|---|---|
| none | The number of in-sequence messages received |
| duplicate messages | The number of duplicate messages received |
| late messages | The number of late message received |
| missed messages | The number of messages missed |
| loss of sync | The number of losses of sync detected |

To determine the size of a reasonable gap in received message numbers, a *guard band* is used. The *guard band* provides a lower and upper bound for received message numbers. It is a configured number in the NMIB, and has a default value of 5.

The following algorithm uses the guard band and the value of the message number in the last delivered message (LDM) to determine the misordering type.

– When a subscriber VCR endpoint receives its first message, it delivers it and saves its message number as the initial value of its LDM.

– Following the receipt of the first message, when a subscriber VCR endpoint receives a new message, it determines the type of misordering, using the modulus of the message number for all calculations, as shown in Table 86.

**Table 86 – Determination of misordering type at a subscriber VCR**

| Misordering Type | Calculation |
|---|---|
| none | Received Message Number = LDM + 1 |
| duplicate messages | Received Message Number = LDM |
| late messages | (LDM – Guard Band) =< Received Message Number < LDM |
| missed messages | LDM + 1 < Received Message Number =< (LDM + Guard Band) |
| loss of sync | (LDM – Guard Band) > Received Message Number OR Received Message Number > (LDM + Guard Band) |

Figure 10 summarizes the misordering message handling.

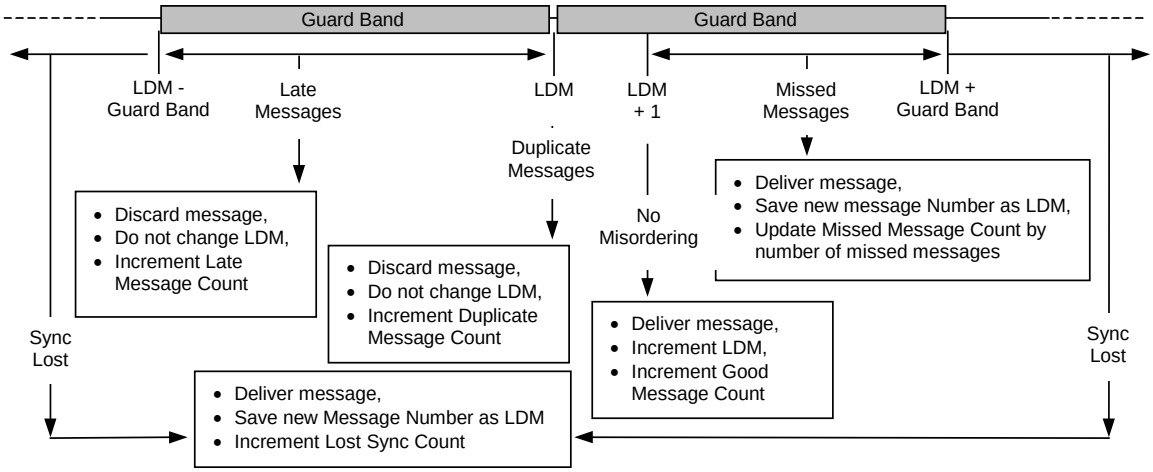



**Figure 10 – Misordered message handling**

### 7.1.5.5.5 Converting messages to service primitives

Table 87 specifies the mapping of the received message type to the appropriate primitive type based on the location of the SMK or VFD being accessed.

**Table 87 – Mapping of received messages to primitives**

| Input Message Type | → | Final Destination | FDA Interface Type | Delivered To |
|---|---|---|---|---|
| SM Request | → | Local SMK | SM Request | Local SMK |
| SM Response | → | Local SMK | SM Response | Local SMK |
| SM Request | → | Type 9 SMK | Request Primitive | Local Type 5 SMK |
| Request Message | → | Local VFD | Indication Primitive | Local VFD |
| Response Message | → | Local VFD | Confirmation Primitive | Local VFD |
| Request Message | → | Type 9 VFD | Request Primitive | Type 9 Comm Stack Interface* |
| Response Message | → | Type 9 VFD | Response Primitive | Type 9 Comm Stack Interface* |
| NOTE For I/O Gateways, the Type 9 Comm Stack interface abstraction is maintained . | | | | |

#### 7.1.5.5.6 Converting service primitives to messages

Table 88 specifies the mapping of the received primitive type to the message type based on the location of the SMK or VFD originating the primitive.

**Table 88 – Mapping of received primitives to messages**

| Requester | FDA Interface Type | → | Output Message Type |
|---|---|---|---|
| Local SMK | Request Primitive | → | Request Message |
| Local SMK | Response Primitive | → | Response Message |
| Local VFD | Request Primitive | → | Request Message |
| Local VFD | Response Primitive | → | Response Message |
| Type 9 Comm Stack Interface* | Indication Primitive | → | Request Message |
| Type 9 Comm Stack Interface* | Confirmation Primitive | → | Response Message |
| NOTE For I/O Gateways, the Type 9 Comm Stack interface abstraction is maintained. | | | |

#### 7.1.5.5.7 Receiving messages

After initialization, the FDA Agent listens to the SM and FDA Ports and to the other configured ports for FDA Messages. It distinguishes and delivers the following types of messages, as shown in Figure 11 through Figure 15.

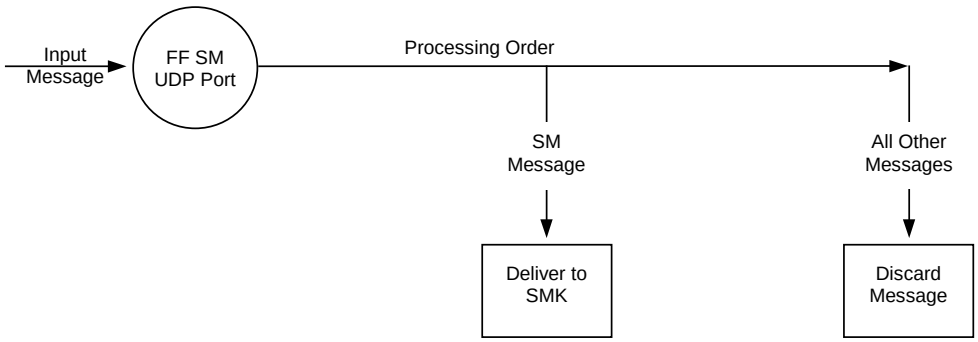
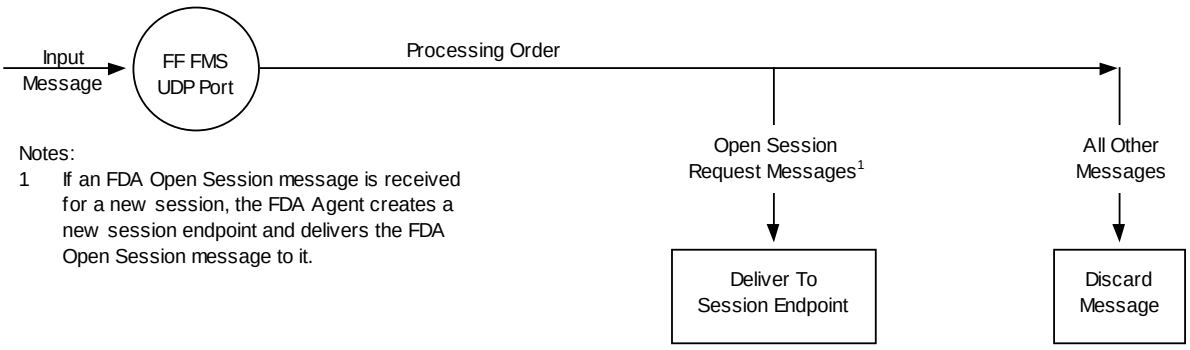



**Figure 11 – FF SM port message processing order**




Notes:
1 If an FDA Open Session message is received for a new session, the FDA Agent creates a new session endpoint and delivers the FDA Open Session message to it.

**Figure 12 – FF FDA port message processing order**

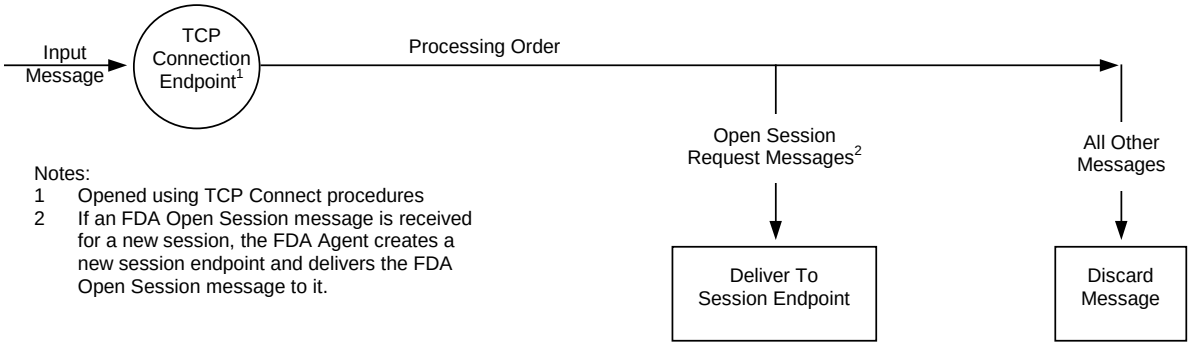



**Figure 13 – FF TCP connection message processing order**

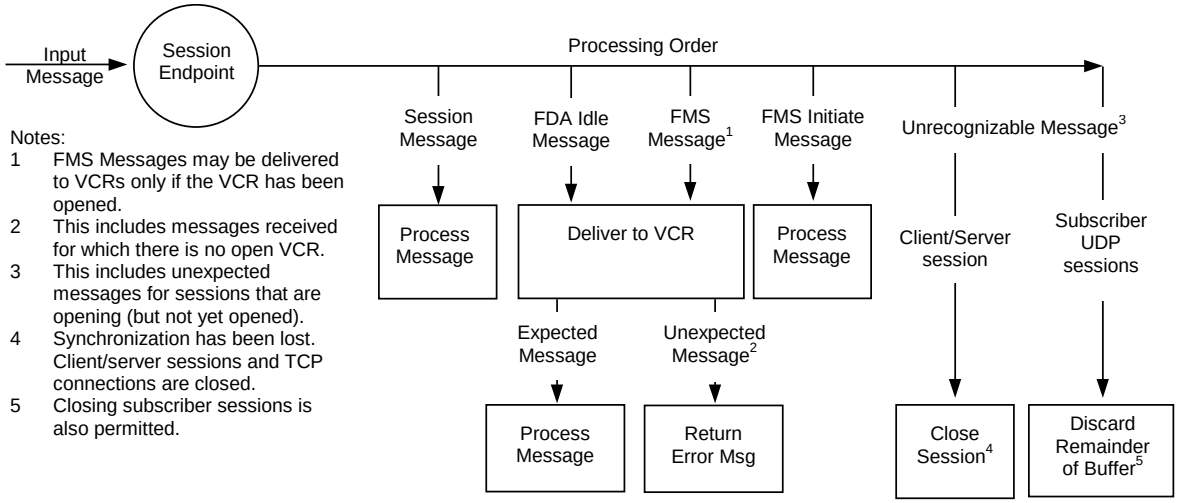



**Figure 14 – Session endpoint message processing order**

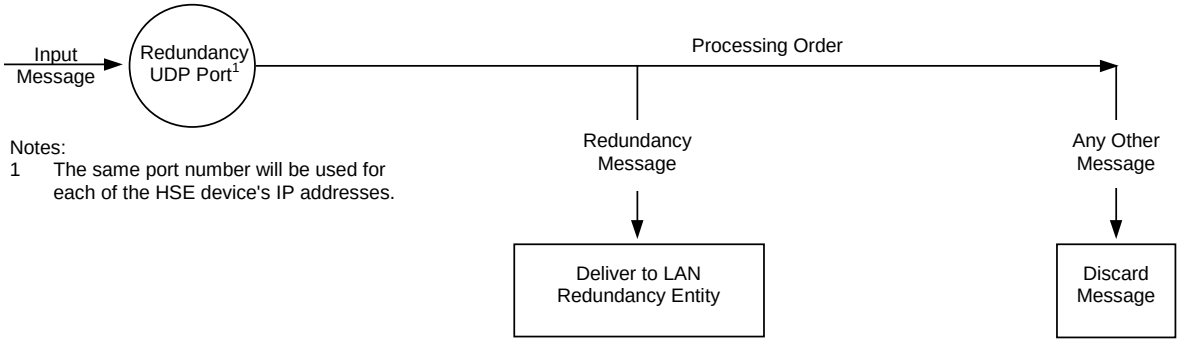
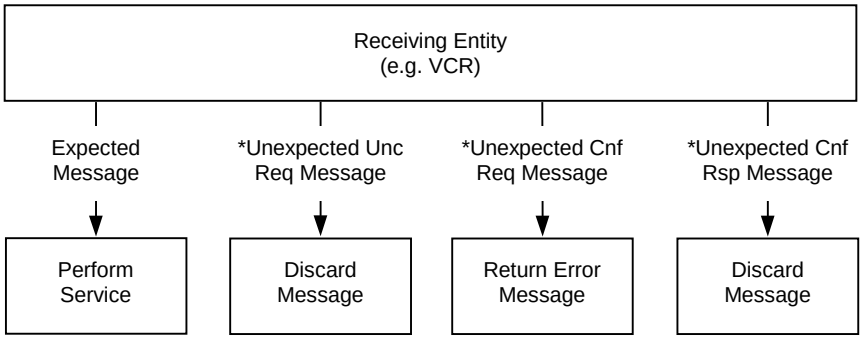



**Figure 15 – FDA LAN redundancy port message processing order**

Figure 16 summarizes the processing of confirmed and unconfirmed messages by the receiving entity.




* An unexpected message is a message that is valid message that is not expected by the entity in its current state.

**Figure 16 – Message processing by receiving entity**

#### 7.1.5.5.8 Starting and stopping timers

##### 7.1.5.5.8.1 VCR idle timer

The VCR Idle Timer is used to keep client/server VCRs open in the absence of user requests. The client VCR endpoint restarts this timer each time it sends a message. Server VCR endpoints do not restart this timer.

##### 7.1.5.5.8.2 Inactivity close timer

The Inactivity Close Timer controls how long client and server VCR and session endpoints remain open. Client and server VCR and session endpoints restart this timer each time they receive a message.

##### 7.1.5.5.8.3 Transmit delay timer

The Transmit Delay Timer controls how long a session endpoint waits after accepting a message for transfer to transfer it. The timer is restarted when the first message is placed into the transmit buffer after the buffer has been initialized or after it has been emptied by the previous transmission. It is stopped when the buffer is filled, causing the buffer to be transmitted and then emptied.

#### 7.1.5.5.9 Message Padding

FDA messages are designed for efficient processing by positioning 16-bit and 32-bit fields on the appropriate word boundaries. The trailer fields, however, are located as an offset from the end of the message. Therefore, when user data is present in a message, the boundary alignment of trailer fields is dependent on the length of the user data. In addition, when messages are concatenated into a single buffer for transfer, the boundary alignment for the FDA Message Header is dependent on the length of the preceding message.

To maintain efficient boundary alignments, an optional padding capability is defined. Using it, an FDA session can be configured using the Message Header Option Bits to insert pad bytes after the user data and before the trailer fields for services that contain an index in the request or corresponding response. To insert bytes, the sending FDA Agent sets the Message Header Options pad bits to indicate how many pad bytes are inserted between the user data and the trailer fields. The value of each pad byte is set to binary 0, and the FDA Message Header Options “Pad Length” is set to indicate the number of pad bytes.

It is a protocol error to insert padding that does not properly align the message trailer to a 4-octet boundary or that does not pad the message length to a multiple of 4 octets. This restriction is designed to support devices that have word sizes of either 4 or 8 octets.

It is a configuration error to connect a session endpoint that sends padded messages to one that is not capable of receiving them.

NOTE 1 The NMIB indicates whether the FDA Agent supports padding as a sender or a receiver.

NOTE 2 LAN Redundancy and SM Messages are not sent within the context of a session, and are therefore, not padded.

#### 7.1.5.6 Identifying Type 5 and Type 9 applications

##### 7.1.5.6.1 General

FDA Messages may be sent or received by either Type 5 or Type 9 applications/SMKs. To identify the sending or receiving application/SMK, a combination of the network address and the FDA Address is used. The FDA Address contains the Link Id and the Selector. Each of these identifiers is discussed below.

##### 7.1.5.6.2 Network addresses and subnet addresses

Network addresses are used as source and destination address parameters for the sending and receiving devices. Source addresses are individual addresses while destination addresses may be individual or multicast addresses. When conveyed within an FDA message, the Network Address uses a 16 octet address format.

Subnet addresses are defined as the first N bits of a device's individual network address, starting with the most significant bit. The number N is configurable and is maintained in the MIB.

For multicast messages, the actual device originating the message is not significant, but the Subnet where the message originated is. Therefore, the Subnet of the sender is used to qualify the FDA Address contained in the message header of multicast messages.

The Network Addresses defined in this specification are listed in Table 89.

**Table 89 – Defined network addresses**

| Address Name | Address Use | Value |
|---|---|---|
| Individual | Address of device interface | Assigned by DHCP |
| Operational Network Address | Address of device interface used in Find Tag Reply messages | Assigned during device assignment |
| SM Multicast | Multicast address of SMKs | Reserved |
| Diagnostic Message Interface A Send Address | Multicast address used to send LAN Redundancy Diagnostic messages from network interface A. | Assigned using the LAN Redundancy Put Information service |
| Diagnostic Message Interface B Send Address | Multicast address used to send LAN Redundancy Diagnostic messages from network interface B. | Assigned using the LAN Redundancy Put Information service |
| Publisher/Report Source multicast address | Multicast address to which publishers and report sources send their messages | Configured in the MIB |

Their use is shown in Table 90.

**Table 90 – Use of network addresses**

| Address Use | Address Type | Used to Identify |
|---|---|---|
| Source | Individual | Message sender's subnet and the Device within the subnet |
| Destination | Individual | Intended receiver's subnet and the Device within the subnet |
| Destination | Multicast | Group of intended receivers |

#### 7.1.5.6.3 The selector

##### 7.1.5.6.3.1 General

The Selector is the low order 16-bits of the FDA Address. It identifies a specific communication endpoint within the identified device or Type 9 Link, or it is used to complete a 32-bit flat address. Each session type uses a different type of selector. The Selector may be hierarchical or flat.

##### 7.1.5.6.3.2 Server selectors

###### 7.1.5.6.3.2.1 General

Server Selectors are used in the FDA Address field of the FDA Message Header in both request and response messages to identify a VCR endpoint used to access the server VFD of a client/server session.

###### 7.1.5.6.3.2.2 Scope

Server VCR Selectors that identify a Type 9 VCR are qualified by the Link Id and are unique within the identified Type 9 link. Server VCR Selectors that identify a Type 5 VCR have a Link Id of 0 and are unique within the device. They are not required to be unique across devices.

###### 7.1.5.6.3.2.3 Use

Server VCR endpoint selectors are used only for Client/Server sessions as shown in Table 91.

**Table 91 – Use of endpoint selectors in server VCRs**

| Session Endpoint Type | Use |
|---|---|
| **Server** | Identifies the server VCR endpoint within a Type 5 or Type 9 device. For Type 5 VCR selectors, two values are used. One value is used in the Initiate Request message to identify a generic server VCR endpoint for the associated VFD. The second is the selector for the VCR endpoint that was dynamically created as a result of processing the Initiate request. This selector is the OD index of the Automatic VCR in the MIB. |

##### 7.1.5.6.3.3 Publisher/subscriber selectors

###### 7.1.5.6.3.3.1 General

Publisher Selectors are used in the FDA Address field of the FDA Message Header to identify the publisher data buffer for publisher/subscriber sessions. The index parameter contained in the message identifies the individual parameter being published.

###### 7.1.5.6.3.3.2 Scope

Publisher Selectors are unique within the Link Id value.

#### 7.1.5.6.3.3.3 Use

Publisher Selectors are used only for Publisher/Subscriber sessions as shown in Table 92.

**Table 92 – Use of endpoint selectors in publisher VCRs**

| Session Endpoint Type | Use |
|---|---|
| **Publisher** | Identifies the publisher data buffer within a Type 5 or a Type 9 device. They may be hierarchical, or flat within the link id value. When hierarchical, the first octet identifies the Type 9 device and the second identifies the DL-connection endpoint within the Type 9 device. |
| NOTE Publisher and subscriber VCR endpoints are stored as static VCRs in the MIB. Their OD Indexes are not used as selector values. | |

### 7.1.5.6.3.4 Report distribution selectors

#### 7.1.5.6.3.4.1 General

Report Source Selectors are used in the FDA Address field of the FDA Message Header to identify the Report Source VFD.

#### 7.1.5.6.3.4.2 Scope

Report Source Selectors are unique within the Link Id value.

#### 7.1.5.6.3.4.3 Use

Report Source Selectors are used only for Report Distribution sessions as shown in Table 93.

**Table 93 – Use of endpoint selectors in source VCRs**

| Session Endpoint Type | Use |
|---|---|
| **Report Source** | Identifies the report source VFD within a device or Type 9 link. They may be hierarchical or flat within the link id value. When hierarchical, the first octet identifies the Type 9 device and the second identifies the DL-service access point within the Type 9 device. |
| NOTE Report source and report sink VCR endpoints are stored as static VCRs in the MIB. Their OD Indexes and are not restricted to 16-bit values, nor are they used as selector values. | |

### 7.1.5.7 Using underlying connectionless and connection-oriented services

#### 7.1.5.7.1 General

Connectionless service use is the default used to support sessions. Connection-oriented services may be used as an option for client/server sessions. The selection of which protocol to use is made by the client or by the configuration application if the client endpoint is configured.

#### 7.1.5.7.2 Connectionless

Connectionless services between two or more FDA Agent session endpoints are message-oriented. Therefore, each message buffer always contains an integral number of FDA messages. FDA messages are never split across message buffers.

Although connectionless services can be used for all types of FDA Agent sessions, they are always used to support FDA Agent sessions that require multicast delivery or that do not require reliable, flow-controlled sequenced message delivery.

#### 7.1.5.7.3 Connection-oriented

Connection-oriented services are used between two and only two FDA Agent session endpoints. Connections are only used to support FDA Agent client/server sessions. That is, they are not used to support the sessionless transfer of SM messages or to support multicast transfers.

Connection-oriented services should be used for FDA Agent sessions that wish to exchange data in a reliable, flow-controlled, and sequenced manner.

#### 7.1.5.7.4 Concatenating messages

Connectionless and connection-oriented services transparently transfer the contents of buffers between applications. Neither inspect the contents or make changes to them.

FDA sessions place their messages into buffers before sending them. The FDA Message structure has been designed to allow sessions to place (concatenate) more than one message into the same buffer for transfer. This can improve network utilization and can reduce the number of interrupts at the receivers. Because FDA SM Messages are not sent on sessions, they are not concatenated into buffers.

The number of messages that can be concatenated together and sent is limited by the size of the sending buffer. Buffers are sent as soon as they are filled. They always are sent with an integral number of messages in them when using underlying connectionless services. When a message is submitted for transfer using underlying connectionless services that would overflow the buffer, the buffer is sent without it, and the message is inserted as the first message in the next buffer to be sent. When underlying connection-oriented services are used, messages may span buffers because the transfer and delivery may be stream oriented.

To prevent buffers from waiting indefinitely to be filled, session endpoints contain an attribute called the Transmit Delay Time. This attribute dictates how long the FDA Agent waits for the buffer to fill. If it is not filled within this time period, the FDA Agent transfers its contents and prepares a new buffer for the next transfer.

#### 7.1.5.7.5 Port numbers

##### 7.1.5.7.5.1 General

Ports identify the access point of an application to the underlying layer. There are three port addresses used by the FDA. One is used exclusively for SM Device Annunciation. It is referred to as the Annunciation Port. The second port is used for SMK access. It is referred to as the SM Port.

The third reserved port is used for receiving FDA Open Session request messages. It is referred to as the FDA Port. Positive responses to Open Session request messages are sent from previously unused ports, and these ports are used to transfer all subsequent session messages. Negative responses to Open Session request messages are sent from the FDA Port.

Data paths between applications are identified by a combination of the source and destination Network Address/Port Number.

##### 7.1.5.7.5.2 Scope

Port numbers are unique within an individual network address. The same port number may be used for connectionless or connection-oriented sockets at the same time without confusion. The underlying protocol qualifies the port number.

#### 7.1.5.7.5.3 Use

FDA Agent Session types use port numbers in conjunction with the network addresses as shown in Table 94 through Table 103. In these tables, the entries in the columns show the addresses to which the endpoint is bound.

**Table 94 – Network address and port numbers for device annunciation**

| Transfer Type | Device SMK annunciates from: | | Configuration AP Listens to: |
|---|---|---|---|
| Multicast | Device Network Address<br>Any Unused Port or the Reserved FDA Port | → | Reserved SM Multicast Network Address<br>Reserved Annunciation Port |

**Table 95 – Network address and port numbers for set/clear assignment info and clear address**

| Transfer Type | Configuration AP sends and receives using: | → | SMK listens to and responds from: |
|---|---|---|---|
| Connectionless Unicast | Configuration AP Network Address<br>Any Unused Port | ← | Device Operational Network Address<br>Reserved SM Port |

**Table 96 – Network address and port numbers for SM identify**

| Transfer Type | Requesting AP Sends Request From: | | Responding SMK: listens to | Responding SMK: responds from |
|---|---|---|---|---|
| Connectionless Unicast | Requester Network Address<br>Any Unused Port | → | Device Operational Network Address<br>Reserved SM Port | Device Operational Network Address<br>Reserved SM Port |
| Multicast | | ← | Reserved SM Multicast Network Address<br>Reserved SM Port | |

**Table 97 – Network address and port numbers for SM find tag**

| Transfer Type | Requesting AP Sends Query From: | | Replying SMK: listens to | Replying SMK: replies from |
|---|---|---|---|---|
| Connectionless Unicast | Requester Network Address<br>Any Unused Port | → | Device Operational Network Address<br>Reserved SM Port | Device Operational Network Address<br>Reserved SM Port |
| Multicast | | ← | Reserved SM Multicast Network Address<br>Reserved SM Port | |

**Table 98 – Network address and port numbers for clients and servers (part 1)**

| Transfer Type | Client AP opens session from: | | Server Device: listens to | Server Device: responds from |
|---|---|---|---|---|
| Connectionless Unicast | Client Network Address<br>Any Unused Port | → | Device Operational Network Address<br>Reserved FDA Port | Device Operational Network Address<br>Any Unused Port (see Note) |
| | | ← | | |

NOTE Also used as the port for receiving and sending all subsequent messages on the session. Negative responses are returned from the FDA Port if the server is unable to acquire a port for responding.

**Table 99 – Network address and port numbers for clients and servers (part 2)**

| Transfer Type | Client AP opens TCP Connection from: | | Server Device: listens to | Server Device: responds using |
|---|---|---|---|---|
| Connection-oriented | Client Network Address<br>Any Unused Port | →<br>← | Device Operational Network Address<br>Reserved FDA Port | Device Operational Network Address<br>The newly established TCP connection (see Note) |
| NOTE The TCP connection is used by both endpoints for sending and receiving all messages on the session, including the FDA Open Session request and response messages. | | | | |

**Table 100 – Network address and port numbers for publishers and subscribers**

| Transfer Type | Publisher device sends from: | | Subscriber device listens to: |
|---|---|---|---|
| Multicast | Publisher Device Operational Network Address<br>Any Unused Port | → | Configured Multicast Network Address<br>Configured Port* |
| * Multicast addresses and Port Numbers are configured as the destination address/port for publisher endpoints. Subscriber endpoints are configured to receive messages at these addresses/ports. The Configured Port is assigned from the unused range of port numbers. Reserved port numbers or port numbers used for other purposes may not be used. | | | |

**Table 101 – Network address and port numbers for report distribution**

| Transfer Type | Report Source device sends from: | | Report Sink device listens to: |
|---|---|---|---|
| Multicast | Report Source Device Operational Network Address<br>Any Unused Port | → | Configured Multicast Network Address*<br>Configured Port* |
| * Multicast addresses and Port Numbers are configured as the destination address/port for report source endpoints. Report sink endpoints are configured to receive messages at these addresses/ports. The Configured Port is assigned from the unused range of port numbers. Reserved port numbers or port numbers used for other purposes may not be used. | | | |

**Table 102 – Network address and port numbers for LAN redundancy get and put information**

| Transfer Type | Client AP opens session from: | | Server Device: listens to | Server Device: responds from |
|---|---|---|---|---|
| Unicast | Client Operational Network Address<br>Any Unused Port | →<br>← | Device Operational Network Address<br>Assigned LAN Redundancy port | Device Operational Network Address<br>Assigned LAN Redundancy port |

**Table 103 – Network address and port numbers for LAN redundancy diagnostics**

| Transfer Type | Device sends from: | | Devices listen to: |
|---|---|---|---|
| Multicast | Device Operational Network Address<br>Any Unused Port | →<br>← | Assigned Multicast Address<br>Assigned LAN Redundancy port |

### 7.1.6 Support for configuration management

Configuration data for the FDA Agent is maintained in the MIB. The FDA Agent also uses the bridge data structures in the MIB if it is in a Linking Device and is not supported by Type 9 bridging capabilities. It uses the bridge data structures to perform republishing and report forwarding operations normally performed by the bridge.

## 7.2 ASEs

### 7.2.1 Virtual field device ASE

#### 7.2.1.1 Overview

Type 5 VFD is the same as the Type 9 VFD except for the VCR object and the Initiate Service. They replace those found in the Type 9 specification. All other Type 9 objects and services apply. They are not repeated here.

#### 7.2.1.2 Simple virtual field device model class specifications

##### 7.2.1.2.1 Class overview

This subclause provides the formal class definition for the VCR class. VCRs are analogous to Type 9 VCRs.

##### 7.2.1.2.2 VCR formal model

The VCR class specifies the attributes and services of the VCR. Its parent class "top" indicates the top of the FAL class tree.

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **VFD ASE** |
| **CLASS:** | | | **VCR** |
| **CLASS ID:** | | | Not used |
| **PARENT CLASS:** | | | TOP |
| **SYSTEM MANAGEMENT ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | VCR Id |
| 2 | (m) | Attribute: | Related AREP Id |
| 3 | (m) | Attribute: | VCR Type |
| 4 | (m) | Attribute: | VCR State |
| 5 | (m) | Attribute: | VCR User Id |
| 6 | (m) | Attribute: | FDA Address |
| 7 | (c) | Constraint | VCR Type = LOCAL SUBSCRIBER<br>\|\| TYPE 9 PUBLISHER<br>\|\| LOCAL REPORT SINK<br>\|\| TYPE 9 REPORT SOURCE |
| 7.1 | (m) | Attribute: | Subnet Mask |
| 9 | (c) | Constraint | VCR Type = LOCAL PUBLISHER |
| 9.1 | (m) | Attribute: | On Change Threshold |
| 7 | (c) | Constraint | VCR Type = LOCAL CLIENT<br>\|\| TYPE 9 SERVER<br>\|\| LOCAL SERVER<br>\|\| TYPE 9 CLIENT<br>\|\| LD TYPE 9 MIB AGENT |
| 7.1 | (m) | Attribute: | Inactivity Close Time |
| **SERVICES:** | | | |
| 1 | (o) | Ops Service: | Initiate |

##### 7.2.1.2.3 System management attributes

The following attributes are accessible through System Management. They are not otherwise accessible.

**Local VCR Id**

This attribute identifies the VCR.

### Related AREP Id

This attribute is the numeric identifier of the related AREPs used to send or receive messages. This value is 0 when the VCR is used to republish or forward reports between two Type 9 ARs in a linking device. The value 0 signifies that there is no related session endpoint.

### VCR Type

This attribute indicates the type of the VCR. Each of the types uses the Related AREP to convey services. Its value are as shown in Table 104.

**Table 104 – VCR types**

| VCR Type | Use |
|---|---|
| LOCAL CLIENT | The VCR receives confirmed request primitives from the AP. |
| LOCAL SERVER | The VCR delivers confirmed indication primitives to the AP for processing. |
| LOCAL PUBLISHER | The VCR publishes data originating in the AP. |
| LOCAL SUBSCRIBER | The VCR subscribes to published data and delivers it to the AP. |
| LOCAL REPORT SOURCE | The VCR distributes reports originating in the AP. |
| LOCAL REPORT SINK | The VCR receives reports and delivers them to the AP. |
| TYPE 9 SERVER[1] | The VCR is a server on a Type 9 link and a client on a Type 5 network. The VCR receives confirmed indication primitives from the Type 9 link and forwards them onto the Type 5 network. |
| TYPE 9 CLIENT[1] | The VCR is a client on a Type 9 link and a server on a Type 5 network. The VCR receives confirmed indication primitives from the Type 5 network and forwards them onto a Type 9 link. |
| TYPE 9 SUBSCRIBER[1] | The VCR is a subscriber on a Type 9 link and a publisher on a Type 5 network. The VCR receives published data from the Type 9 link and republishes it onto the Type 5 network. |
| TYPE 9 PUBLISHER[1] | The VCR is a publisher on a Type 9 link and a subscriber on a Type 5 network. The VCR receives published data from the Type 5 network and republishes it onto the Type 9 link. |
| TYPE 9 REPORT SINK[1] | The VCR is a report sink on a Type 9 link and a report source on a Type 5 network. The VCR receives reports from the Type 9 link and forwards them onto the 5 network. |
| TYPE 9 REPORT SOURCE[1] | The VCR is a report source on a Type 9 link and a report sink on a Type 5 network. The VCR receives reports from the Type 5 network and forwards them onto the Type 9 link. |
| LD TYPE 9 MIB AGENT[1] | The VCR delivers confirmed indication primitives to the Type MIB Agents for processing (applicable only to linking devices). |

NOTE Type 9 VCRs are present only in Linking Devices.

### VCR State

This attribute specifies the state of the VCR. Valid states are defined in IEC 61158-6-5.

### VCR User Id

This conditional attribute is the FDA Address that identifies the user of the VCR. The use of this attribute is shown in Table 105.

**Table 105 – Use of VCR user id**

| VCR Type | Contents (represented as an FDA Address): |
|---|---|
| LOCAL SERVER | The FDA Address contained in the Type 9 Initiate Request used to open the VCR. |
| LD TYPE 9 MANAGEMENT AGENT | Type 9 link of the Management Agent. The N.S portion of the FDA Address is unused and set to 0. |
| TYPE 9 SERVER | Type 9 Server VCR used to receive requests from the Type 9 link. |
| TYPE 9 CLIENT | Type 9 Client VCR used to send requests on the Type 9 link. |
| TYPE 9 SERVER | Type 9 Server VCR used to receive requests from the Type 9 link. |
| TYPE 9 PUBLISHER | Type 9 Publisher VCR used for publishing onto the Type 9 link. |
| TYPE 9 SUBSCRIBER | Type 9 Subscriber VCR used to subscribe to data on the Type 9 link. |
| TYPE 9 REPORT SOURCE | Type 9 Report Source VCR used for sending reports onto the Type 9 link |
| TYPE 9 REPORT SINK | Type 9 Report Sink VCR used to receive report distributions from the Type 9 link. |

### FDA Address

This attribute specifies a reference that is conveyed with each service to or from the corresponding VCR endpoint(s). Its values are VCR Type dependent as shown in Table 106.

**Table 106 – Use of FDA address**

| VCR Type | Use |
|---|---|
| LOCAL CLIENT | • The Id of the associated server VCR or Type 9 AREP. |
| LOCAL SERVER | • The VCR Id of the LOCAL SERVER VCR endpoint. |
| LOCAL PUBLISHER | • "Published-Data" reference used to identify the published data buffer.<br>• This address is taken from the 32-bit Type 9 flat DLCEP address space.<br>• The Type 5 Subnet Mask of the publisher qualifies this "Published-Data" reference. |
| LOCAL SUBSCRIBER | • The reference for the publisher data buffer that is "Subscribed-To".<br>• The Type 5 Subnet Mask of the publisher qualifies this "Subscribed-To" reference. |
| LOCAL REPORT SOURCE | • "Reporting VFD" reference used to identify the VFD sending the report messages.<br>• This address is taken from the 32-bit Type 9 flat DLSAP address space.<br>• The Type 5 Subnet Mask qualifies this "Reporting VFD" reference.<br>• This VFD identifier qualifies the index parameter value contained in the report messages in the service-specific parameters (each source VFD may send one or more report messages, each with a different Index). |
| LOCAL REPORT SINK | Not Used. A single LOCAL REPORT SINK VCR may receive reports from more than one report source. Therefore, the APDUs received from the related AREP may contain a different FDA Address value. |
| TYPE 9 SERVER | • The numeric id of the Type 9 Server AREP. |
| TYPE 9 CLIENT | Not Used. |
| TYPE 9 PUBLISHER | • If the Related AREP Id is non-zero, then this is the *FDA Address* in received Type 5 publisher APDUs that is "Subscribed-To". The Type 5 Subnet Mask of the publisher qualifies this "Subscribed-To" reference.<br>• If the Related AREP Id is zero, then this value is the Type 9 DLCEP used by the Type 9 Subscriber VCR to receive published data.<br>• The received data is republished onto the Type 9 link using the VCR User Id attribute. |
| TYPE 9 SUBSCRIBER | • Type 9 data link address of the published data buffer. |
| TYPE 9 REPORT SOURCE | • Not Used. A single TYPE 9 REPORT SOURCE VCR may receive reports from more than one report source. Therefore, the APDUs received from the related AREP may contain a different FDA Address value. |
| TYPE 9 REPORT SINK | • Not Used. A single TYPE 9 REPORT SINK VCR may receive reports from more than one report source. Therefore, the reports received from the Type 9 VCR may contain a differentType 9 source address. |

**Type 5 Subnet Mask**

This attribute identifies the subnet of FDA Address and is used to qualify it for service indication primitives received on the VCR.

**On Change Threshold**

This conditional attribute is present only in LOCAL PUBLISHER VCRs. It specifies the percentage of change to be exceeded before the value of the published data is considered to be changed. If its value is 0, then every value is published using the Type 9 Information Report service, whether or not it has changed. The default value is 0.

If its value is not 0, then changed values are published using the Type 5 Information Report On Change service. The On Change capability operates as follows. The acceptable difference is calculated by multiplying the published data range, specified by the application, by the On Change Threshold percent. If the data range between high and low values were 20 and the On Change Threshold were 10%, the acceptable difference would be 2. When a published variable value is available, the absolute value of its difference with the last published value is calculated. If the new value is 17 and the previous value was 14, then the difference is 3. If this difference is greater than the acceptable difference (3 is greater than 2), then the value is considered to have changed and is submitted for transfer.

If the value has not been transferred for ”refresh rate” number of publisher execution cycles, the latest value is transferred. The refresh rate attribute is contained in the MIB. Support for the On Change capability is optional, although support for this attribute is mandatory.

NOTE Whether this capability is implemented is not specified. If it is implemented in the AE, then the AE needs to be able to decode the data portion of the message to determine the value that is being published. If the application implements this capability, then it needs to read the value of this attribute to determine whether or not to submit the data for publishing.

**Inactivity Close Time**

This attribute controls how long a client VCR endpoint remains open without receiving a message and it also specifies the time interval that the client VCR endpoint uses to control the sending of Idle messages. Its value is inherited from its related AREP.

#### 7.2.1.2.4 Services

**Initiate**

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

### 7.2.1.3 Virtual field device ASE service specification

#### 7.2.1.3.1 Services supported

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

– Initiate

#### 7.2.1.3.2 Initiate service

##### 7.2.1.3.2.1 Service overview

This service is used to open to remote VCR endpoint. The remote endpoint can be in a Type 5 device or in a Type 9 device that is connected to the Type 5 device by a linking device.

##### 7.2.1.3.2.2 Service primitives

The service parameters for this service are shown in Table 107.

**Table 107 – Initiate**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| FDA Address | M | M | | |
| Connect Option | U | | | |
| Access Protection Supported Calling | M | M (=) | | |
| Password Calling | M | M (=) | | |
| Access Groups Calling | M | M (=) | | |
| Version OD Calling | M | M (=) | | |
| Profile Number Calling | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| VCR Id | | | C | C (=) |
| Version OD Called | | | M | M (=) |
| Profile Number Called | | | M | M (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| VCR Id | | | C | C (=) |
| Error Class and Code | | | M | M (=) |
| Additional Code | | | U | U (=) |
| Additional Description | | | U | U (=) |

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

### Argument

This parameter carries the parameters of the service invocation.

### FDA Address

This parameter identifies the remote VCR endpoint. Its value is dependent on the value of the Connect Option

### Connect Option

This parameter indicates which option is to be used to open the VCR. Its values are listed in Table 108.

**Table 108 – Connect option**

| Connect Option | Value | Description |
|---|---|---|
| VCR Selector | 1 | This option allows the requesting application to provide the VCR Selector that is to be used to connect to the VFD. If the VFD is a Type 5 VFD, then the VCR Selector is a generic selector for accessing the VFD. If the VFD is a Type 9 VFD, then the VCR Selector is the Id of the Type 9 Client AREP to be used to access the Type 9 device. This Type 9 Client AREP cannot be associated with the standard MIB access VCR endpoint in the Type 9 device. |
| MIB [1] | 2 | This option allows the requesting application to indicate that it desires to connect to a Type 9 or Type 5 MIB VFD. The link id portion of the FDA Address field indicates whether or not the MIB is in the Type 9 or Type 5 device. |
| FBAP [2] | 3 | This option allows the requesting application to indicate that it desires to connect to a Type 9 or Type 5 Function Block Application Process (FBAP) VFD. The link id portion of the FDA Address field indicates whether or not the MIB is in the Type 9 or Type 5 device. The node id portion of the FDA Address field identifies the node containing the FBAP, and the selector portion of the FDA Address field is an ordinal number that starts with "1" to indicate which FBAP is to be accessed |
| NOTE 1 The MIB can only be accessed using this Connect.<br>NOTE 2 The FBAP can be accessed also using the VCR Selector Connect Option. | | |

### Access Protection Supported Calling

This parameter specifies the value of the calling VFD'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 VCR. If access with password is not to be used on this VCR, 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 VCR.

### Version OD Calling

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

### Profile Number Calling

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

### Physical Device Tag

This parameter specifies the value of the SMK Physical Device Tag of the device that contains the called VFD if it has one. If it does not, its value is null.

### Result(+)

This selection type parameter indicates that the service request succeeded.

### VCR Id

This conditional attribute identifies the VCR endpoint from which the response was sent.

### Version OD Called

This parameter should specify the version of the server's OD.

### Profile Number Called

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

**Result(–)**

This selection type parameter indicates that the service request failed.

**VCR Id**

This conditional attribute identifies the VCR endpoint from which the response was sent. It is present when the underlying AREP supports multiple concurrent open VCRs.

**Error Class and Code**

This parameter should provide the reason for the failure.

**Additional Code**

This optional parameter provides an additional code that is associated with the error.

**Additional Description**

This optional parameter provides 16 octets of additional information that is associated with the error.

#### 7.2.1.3.2.3 Service procedure

This service operates through a queue. The Confirmed Service Procedure specified in 4.6 applies to this service. The service specific processing is as follows:

If the receiving FDA Agent is not able to locate the VCR identified by the FDA Address, or if the VCR is not AP SERVER or TYPE 9 CLIENT, or if the Physical Device Tag in the request does not match the Physical Device Tag of the device's SMK, the FDA Agent returns a negative response containing the FDA Address from the request.

If the request identifies a local VFD, the FDA Agent constructs a new LOCAL SERVER VCR endpoint and delivers the request to it for processing.

If the request identifies a TYPE 9 CLIENT, the FDA Agent constructs a new local TYPE 9 CLIENT and associates it with the Type 9 Client AREP. If the Type 9 Client AREP is not already open, the FDA Agent opens it using the Type 9 Initiate AP ASE service. This permits more than one TYPE 9 CLIENT to be associated with the Type 9 Client AREP.

If the request accepted, the response is returned to the client by the FDA Agent contains the VCR Id of the newly created local VCR. This Id is to be used by the client for subsequent accesses on the VCR.

If the request is rejected, the FDA Agent deletes the VCR object if it created one, and returns a negative response.

If the VCR was opened to a MIB on an AR that does not support update requests, the server FDA Agent issues a negative response if it receives an update request from the client without delivering the service indication to the System Management MIB.

### 7.2.2 System management kernel ASE

#### 7.2.2.1 Overview

The FAL system management kernel model defines an object that supports a system management agent in device. This system management entity is referred to as the System Management Kernel (SMK) because it provides basic capabilities to add or delete the device from the network and to locate objects within the device.

### 7.2.2.2 FAL system management kernel class specification

#### 7.2.2.2.1 System management kernel formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **SYSTEM MANAGEMENT KERNEL ASE** |
| **CLASS:** | | | **SYSTEM MANAGEMENT KERNEL** |
| **CLASS ID:** | | | Not used |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Key Attribute: | Device Id |
| 2 | (m) | Attribute: | Physical Device Tag |
| 3 | (m) | Attribute: | SMK State |
| 4 | (m) | Attribute: | Duplicate Detection State |
| 5 | (m) | Attribute: | LAN Redundancy Socket Address |
| 6 | (m) | Attribute: | Operational Network Address |
| 7 | (m) | Attribute: | Annunciation Repeat Time |
| 8 | (m) | Attribute: | Device Type |
| 9 | (m) | Attribute: | Device Redundancy State |
| 10 | (m) | Attribute: | Device Index |
| 11 | (m) | Attribute: | Max Device Index |
| 12 | (m) | Attribute: | SMK Attributes Version Number |
| 13 | (m) | Attribute: | Device Version Number |
| 14 | (m) | Attribute: | Operational Powerup |
| **SERVICES:** | | | |
| 1 | (m) | OpsService: | Find Tag Query |
| 2 | (m) | OpsService: | Find Tag Reply |
| 3 | (m) | OpsService: | Identify |
| 4 | (m) | OpsService: | Device Annunciation |
| 5 | (m) | OpsService: | Set Assignment Info |
| 6 | (m) | OpsService: | Clear Assignment Info |
| 7 | (m) | OpsService: | Clear Address |

#### 7.2.2.2.2 Attributes

**Device Id**

This key attribute specifies the vendor specific identification for the device.

**Physical Device Tag**

This attribute specifies the site administered name (tag) assigned to the device.

**SMK State**

The attribute specifies the operational state of the SMK. Its values are:

– No Tag

**Duplicate Detection State**

The attribute specifies whether or not the SMK has detected another device with its Physical Device Tag (PD Tag) or its Device Index. Its values are:

– No Duplicates Detected
– Duplicate Pd Tag Detected
– Duplicate Device Index Detected
– Duplicate Pd Tag and Device Index Detected

**LAN Redundancy Socket Address**

The attribute specifies the socket address used by the device to receive LAN Redundancy APDUs.

**Operational Network Address**

This attribute specifies the network address used by the device. More than one address is used when Local Area Network (LAN) redundancy is supported by the device. When more than one address is supported, only one is designated as the Operational Network Address.

**Annunciation Repeat Time**

This attribute specifies the milliseconds between annunciation messages. Its default value is 15 000 (15 s).

**Device Type**

This attribute specifies the type of device and its redundancy capability. The list of valid types includes Type 9 Linking Device, Type 9 Device, Type 5 Device, Gateway Device.

The device's redundancy capability specifies the device's ability to operate as one of a set of redundant devices. All devices in the set are required to have the same value of this attribute configured. Three values are possible, None, External Control, and Automatic.

External Control indicates that the devices in the set are capable of operating as either a primary or a secondary (backup. An external agent is required to synchronise device data, detect a failure of the primary, and after detection, issue a Set Assignment Info service request to one of the secondaries to change its Device Redundancy State from secondary to primary.

Automatic indicates that the device is capable of synchronising with its redundant partners, detecting a fault in the primary, and automatically selecting a new primary to survive the fault.

**Device Redundancy State**

This attribute specifies the actual device redundancy state of the device. If the device is not participating in redundancy, its value is None. If it is participating as an external controlled device, then its value is either "external control – primary" or "external control – secondary". If it is participating as an automatic device, then its values are "automatic – primary" or "automatic – secondary". This last value indicates that the devices in the set are to select which device is the primary on their own. The Set Assignment Info service can be used to assign the value of this attribute. The assigned value should be compatible with the Device Type attribute.

**Device Index**

This attribute is a site administered number that identifies the device. This index is unique within an FAL subnet.

**Max Device Index**

This attribute defines the maximum value that the device index may have. This effectively indicates the maximum number of devices that may be configured into an FAL subnet. This number is a multiple of 32.

**SMK Attributes Version Number**

This attribute is the version number for the attribute values that precede this attribute. Each times the preceding attribute’s values change, this version number is incremented by 1. This version number is only changed if the value of one of the covered attributes changes. If the device initializes on powerup, the value of the version number is set to zero. If the device supports Operational Powerup and it is true, the annunciation version number may remain the same if the attributes it covers go through the power cycle without changing. The value of SMK State may change if the device supports time synchronization and the status of time sync changes, which causes the value of the annunciation version number to change.

**Device Version Number**

This attribute is the version number that represents the composite state of the static information in the applications in the device. Each time there is change to the static data of any application in the device, this version number is incremented by 1. Its value initializes to zero if the device is initialized. How the SMK determines that a change has occurred is a local matter.

**Operational Powerup**

This boolean attribute indicates when true that the SMK data is to be stored in non-volatile memory so that is is retained through a power cycle.

#### 7.2.2.2.3 Services

All services defined for this class are mandatory.

**Find Tag Query**

This unconfirmed service is to search for an object in the network.

**Find Tag Reply**

This unconfirmed service is used as the reply to the Find Tag Query service. It returns a list of VCR Identifiers that can be used in the Initiate service to access the queried object.

**Identify**

This confirmed service is used to request the SMK to identify itself.

**Device Annunciation**

This unconfirmed service is used to periodically announce the presence of the SMK on the network.

**Clear Address**

This unconfirmed service is used to request an SMK to clear its network address.

**Set Assignment Info**

This confirmed service is used to set a selected subset of the SMK attribute values.

**Clear Assignment Info**

This confirmed service is used to clear a selected subset of the SMK attribute values.

### 7.2.2.3 System management kernel ASE service specification

#### 7.2.2.3.1 Services supported

SMK Services operate directly over socket services. They do not use ARs.

#### 7.2.2.3.2 Find tag query service

##### 7.2.2.3.2.1 Service overview

This unconfirmed service is used to send a query to one or more SMKs to locate an object on the network.

##### 7.2.2.3.2.2 Service parameters

The service parameters for this service are shown in Table 109.

**Table 109 – Find tag query service parameters**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| Invoke ID | U | U (=) |
| Source Address | | M |
| Destination Address | M | M (=) |
| SMK ID | M | M (=) |
| Query Type | M | M (=) |
| Physical Device Tag | C | C (=) |
| VFD Tag | C | C (=) |
| Application Object Tag | C | C (=) |
| Element ID | C | C (=) |
| VFD Reference | C | C (=) |
| Device Index | C | C (=) |

### Argument

The argument contains the parameters of the service request.

### Source Address

This parameter is the address from which the service request was sent. It is used by the responder when returning the reply.

### Destination Address

This parameter is the address to which the service request is to be sent. This address may be an Individual Device Address or the SMK Multicast Address.

### SMK ID

This parameter identifies which SMKs are to respond to the query. The possible values are dependent on the value of the Destination Address. They are as shown in Table 110.

**Table 110 – SMK IDs**

| Destination Address Value | SMK Id Value |
|---|---|
| Individual Device Address | Device SMK |
| Individual Device Address | Device SMK and all Type "X" SMKs attached to the addressed device (does not include the Type “X” SMKs contained in the device) |
| Individual Device Address | Device SMK and all Type "X" SMKs on a specific Type "X" link attached to the addressed device (does not include the Type “X” SMKs contained in the device) |
| Individual Device Address | Specific Type "X" SMK accessible through the addressed device |
| SMK Multicast Address | All SMKs |
| SMK Multicast Address | All SMKs and all Type "X" SMKs (does not include the Type “X” SMKs contained in the device) |
| SM Multicast Address | Device SMK and all Type "X" SMKs on a specific Type "X" link attached to the addressed device (does not include the Type “X” SMKs contained in the device) |

### Query Type

This parameter selects the type of query. The following types are mutually exclusive:

– PD Tag query, Primary device
– PD Tag query, Secondary device
– VFD Tag query (also requires PD Tag)
– Application Object Tag query

– Element ID query (also requires Application Object Tag)
– VFD Reference query (also requires PD Tag)
– Device Index query

**Physical Device Tag**

This conditional type parameter, when present, contains the Physical Device Tag, indicating that the query is to locate a physical device. It is present when the Query Type indicates a Physical Device Tag query or a VFD Tag query.

**VFD Tag**

This conditional type parameter, when present, contains the VFD Tag, indicating that the query is to locate a VFD. It is present when the Query Type indicates VFD Tag query.

**Application Object Tag**

This conditional type parameter, when present, contains the tag of an application object, indicating that the query is to locate a tagged object. It is present when the Query Type indicates Application Object Tag query or Element ID query.

**Element ID**

This conditional parameter, when present, is used to locate an element of an Application Object Tag. When present, the Application Object Tag is also present. It contains a reference to an element of the tagged application object. It is present when the Query Type indicates Element ID query.

**VFD Reference**

This conditional type parameter, when present, contains a numeric reference to an VFD, indicating that the query is to locate a VFD. It is present when the Query Type indicates VFD Reference query or a VFD Tag query.

**Device Index**

This conditional type parameter, when present, contains the Device Index, indicating that the query is to locate a device by its Device Index. It is present when the Query Type indicates Device Index query.

#### 7.2.2.3.2.3 Service procedure

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

### 7.2.2.3.3 Find tag reply service

#### 7.2.2.3.3.1 Service overview

This unconfirmed service is used to send a reply to the initiator of a Find Tag Query. It contains information that can be used to establish an AR and a VCR to access the queried object.

#### 7.2.2.3.3.2 Service parameters

The service parameters for this service are shown in Table 111.

**Table 111 – Find tag reply service parameters**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| Invoke ID | U | U (=) |
| Source Address | | M |
| Destination Address | M | M (=) |
| Duplicate Detection State | M | M (=) |
| Query Type | M | M (=) |
| Queried Object Network Address | M | M (=) |
| Queried Object Link Address | M | M (=) |
| Queried Object Type "X" Device Address | C | C (=) |
| Queried Object Device ID | M | M (=) |
| Queried Object PD Tag | M | M (=) |
| Queried Object VFD Reference | C | C (=) |
| Queried Object OD Version Number | C | C (=) |
| Queried Object Numeric Id | C | C (=) |
| Duplicate Detection State | M | M (=) |
| List of VCR References | C | C (=) |

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which this service was sent.

**Destination Address**

This parameter is the address from which the matching Find Tag Query was sent.

**Duplicate Detection State**

This parameter specifies the value of the SMK attribute of the same name. It refers to the SMK of the replying device.

**Query Type**

This parameter selects the type of query. The following types are mutually exclusive:

– PD Tag query, Primary device
– PD Tag query, Secondary device
– VFD Tag query (also requires PD Tag)
– Function Block Tag query
– Element ID query (also requires FB Tag)
– VFD Reference query (also requires PD Tag)
– Device Index query

**Queried Object Network Address**

This parameter specifies the network address used to access the queried object.

**Queried Object Address**

This parameter is the link id portion of the address to be used to access the queried object. If the queried object is in a device, then this value is 0. If the queried object is located in a Type "X" device on a Type "X" link that is connected to the responding device, then this value is the Type “X” link address.

**Queried Object Type "X" Device Address**

This conditional parameter is present if the queried object is located in a Type "X" device on a Type "X" link that is connected to the responding device. It contains the Type "X" device address used to access the Type "X" device.

**Queried Object Device ID**

This parameter specifies the value of the SMK Device ID attribute for the device that contains the queried object.

**Queried Object PD Tag**

This parameter specifies the value of the SMK PD Tag attribute for the device that contains the queried object.

**Queried Object VFD Reference**

This conditional type parameter, when present, contains a numeric reference of the queried object's VFD. It is present for all queries except for Physical Device Tag queries.

**Queried Object OD Version Number**

This conditional type parameter, when present, contains the version number of the queried object's OD. It is present for all queries except for Physical Device Tag queries.

**Queried Object Numeric Id**

This conditional type parameter, when present, contains the numeric id or OD index attribute of the queried object. It is present if the queried object is contained in a VFD.

**Duplicate Detection State**

This parameter specifies the value of the SMK attribute of the same name.

**List of VCR References**

This conditional type parameter, when present, contains a list of VCR References that can be used with the Type 9 Initiate service to open a VCR to the queried object. It is present for all queries except for Physical Device Tag queries.

#### 7.2.2.3.3.3 Service procedure

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

### 7.2.2.3.4 Identify service

#### 7.2.2.3.4.1 Service overview

This confirmed service is used to obtain identification information from a remote SMK.

#### 7.2.2.3.4.2 Service parameters

The service parameters for this service are shown in Table 112.

**Table 112 – Identify service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| SMK ID | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | M | M (=) |
| Device ID | | | M | M (=) |
| Physical Device Tag | | | M | M (=) |
| SMK State | | | M | M (=) |
| Device Type | | | M | M (=) |
| LAN Redundancy Socket Address | | | C | C (=) |
| Annunciation Repeat Time | | | C | C (=) |
| Device Redundancy State | | | C | C (=) |
| Duplicate Detection State | | | C | C (=) |
| Device Index | | | C | C (=) |
| Max Device Index | | | C | C (=) |
| Operational Network Address | | | C | C (=) |
| SMK Attributes Version Number | | | C | C (=) |
| Device Version Number | | | C | C (=) |
| Version Number List | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Error Info | | | M | M (=) |

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

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the request was sent.

**Destination Address**

This parameter is the address to which the request is sent.

**SMK ID**

This parameter identifies the SMK that is to respond to the request. The responder may be:

– a Type 5 SMK, or
– a Type "X" SMK of a Type "X" link interface in a Type 5 linking device, or
– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**SMK ID**

This parameter identifies the SMK sending the response. The responder may be:

– a Type 5 SMK, or
– a Type "X" SMK of a Type "X" link interface in a Type 5 linking device, or
– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

**Device ID**

This parameter specifies the value of the SMK attribute of the same name.

**Physical Device Tag**

This parameter specifies the value of the SMK attribute of the same name.

**SMK State**

This parameter specifies the value of the SMK attribute of the same name.

**Device Type**

This parameter specifies the value of the SMK attribute of the same name.

**LAN Redundancy Socket Address**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Annunciation Repeat Time**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Device Redundancy State**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Duplicate Detection State**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Device Index**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Max Device Index**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Operational Network Address**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**SMK Attributes Version Number**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Device Version Number**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the responder is a Type 5 device.

**Version Number List**

This conditional parameter is used by device that connects links of two or more Communication Model Types together. It is present if the responder is a Type 5 device.

If the Type "X" address request parameter is not present, then this parameter specifies the link id and version number of each attached link. This Type "X" link version number is incremented each time any device address is added or deleted from the Type "X" network.

If the Type "X" address request parameter identifies a specific link connected to the receiving device, then this parameter specifies the Type "X" address and version number associated with each device connected to that link. This Type "X" device address version number is incremented each time that specific device address is added or deleted from the Type "X" network.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**SMK ID**

This parameter identifies the SMK sending the response. The responder may be:

– a Type 5 SMK, or
– a Type "X" SMK of a Type "X" link interface in a Type 5 linking device, or
– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

#### 7.2.2.3.4.3 Service procedure

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

### 7.2.2.3.5 Device annunciation service

#### 7.2.2.3.5.1 Service overview

This unconfirmed service is used to multicast a notification onto the network that contains basic SMK information. The reserved Annunciation multicast address is used.

#### 7.2.2.3.5.2 Service parameters

The service parameters for this service are shown in Table 113.

**Table 113 – Annunciate service parameters**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| Source Address | | M |
| SMK ID | M | M (=) |
| Device ID | M | M (=) |
| Physical Device Tag | M | M (=) |
| SMK State | M | M (=) |
| Device Type | M | M (=) |
| LAN Redundancy Socket Address | M | M (=) |
| Annunciation Repeat Time | M | M (=) |
| Device Redundancy State | M | M (=) |
| Duplicate Detection State | M | M (=) |
| Device Index | M | M (=) |
| Max Device Index | M | M (=) |
| Operational Network Address | M | M (=) |
| SMK Attributes Version Number | M | M (=) |
| Device Version Number | M | M (=) |
| Version Number List | C | C (=) |

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the request was sent.

**Device ID**

This parameter specifies the value of the SMK attribute of the same name.

**Physical Device Tag**

This parameter specifies the value of the SMK attribute of the same name.

**Device Type**

This parameter specifies the value of the SMK attribute of the same name.

**LAN Redundancy Addr**

This parameter specifies the value of the SMK attribute of the same name.

**Annunciation Repeat Time**

This parameter specifies the value of the SMK attribute of the same name.

**SMK State Capability**

This parameter specifies the value of the SMK attribute of the same name.

**Device Redundancy State**

This parameter specifies the value of the SMK attribute of the same name.

**Duplicate Detection State**

This parameter specifies the value of the SMK attribute of the same name.

**Operational Network Address**

This parameter specifies the value of the SMK attribute of the same name.

**Device Index**

This parameter specifies the value of the SMK attribute of the same name.

**Max Device Index**

This parameter specifies the value of the SMK attribute of the same name.

**SMK Attributes Version Number**

This parameter specifies the value of the SMK attribute of the same name.

**Device Version Number**

This parameter specifies the value of the SMK attribute of the same name.

**Version Number List**

This conditional parameter is present if the service was requested by a device that connects links of two or more Communication Model Types together and if the Type "X" address request parameter is not present. This parameter specifies the version number of each attached link.

#### 7.2.2.3.5.3 Service procedure

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

### 7.2.2.3.6 Set assignment info service

#### 7.2.2.3.6.1 Service overview

This service is used to update the attribute values of the SMK addressed by the Destination Address parameter. The SMK receiving the new values updates its attributes, if possible, and returns a response that includes the newly updated attribute values. This service also may be used to assign a new PD Tag and address to a Type "X" device that is connected to a Type 5 device though a Type "X" network.

#### 7.2.2.3.6.2 Service parameters

The service parameters for this service are shown in Table 114.

**Table 114 – Set assignment info service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| SMK ID | M | M (=) | | |
| Device ID | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| LAN Redundancy Socket Address | C | C (=) | | |
| Device Redundancy State | C | C (=) | | |
| Clear Duplicate Detection State | C | C (=) | | |
| Operational Network Address | C | C (=) | | |
| Device Index | C | C (=) | | |
| Max Device Index | C | C (=) | | |
| Annunciation Repeat Time | C | C (=) | | |
| New Type "X" Address | C | C (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Annunciation Repeat Time | | | C | C (=) |
| Max Device Index | | | C | C (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Error Info | | | M | M (=) |

NOTE See the Note in 3.8.4.3.

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the service was requested.

**Destination Address**

This parameter is the address to which the request is sent.

**SMK ID**

This parameter identifies the SMK that is to respond to the request. The responder may be:

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

**Device ID**

This parameter specifies the value of the SMK attribute of the same name. This value should match that of the SMK attribute and is not writable.

**Physical Device Tag**

This parameter specifies the value of the SMK attribute of the same name. This value is writable, but only when the device does have a Physical Device Tag. Physical Device Tags can be cleared using the Clear Assignment Info service.

**LAN Redundancy Socket Address**

This conditional parameter specifies the value of the SMK attribute of the same name. This value is writable. It is present if the request is addressed to a Type 5 SMK.

**Device Redundancy State**

This conditional parameter specifies the value of the SMK attribute of the same name. This value is writable. It is present if the request is addressed to a Type 5 SMK.

**Clear Duplicate Detection State**

This conditional parameter is used to set the value of the SMK Duplicate Detection State attribute to "Duplicate Physical Device Tag Not Detected" and/or "Duplicate Device Index Not Detected". It is present if the request is addressed to a Type 5 SMK.

**Operational Network Address**

This conditional parameter specifies the value of the SMK attribute of the same name. This value is writable. It is present if the request is addressed to a Type 5 SMK.

**Device Index**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the request is addressed to a Type 5 SMK.

**Max Device Index**

This conditional parameter specifies the value of the SMK attribute of the same name. It is present if the request is addressed to a Type 5 SMK. The responder may negotiate this value to a smaller value, but no smaller than 500. That is, a value smaller than 500 can be returned only if that value or a smaller value is received in request.

**Annunciation Repeat Time**

This conditional parameter specifies the value of the SMK attribute of the same name. This value is writable. It is selected if the request is addressed to a Type 5 SMK. During device assignment, the FDA Agent may negotiate this value to a larger value no larger than 5000 if the assigned value is less than 5000 and the device is not capable of supporting the assigned value.

**New Type "X" Address**

This conditional parameter specifies a NULL value or an address that is to be assigned to a Type "X" device connected to the Type 5 device through a Type "X" network. It is present if the request is addressed to a Type “X” SMK.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**SMK ID**

This parameter identifies the SMK sending the response. The responder may be

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

**Annunciation Repeat Time**

This conditional parameter specifies the negotiated value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

**Max Device Index**

This conditional parameter specifies the negotiated value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Type "X" Address**

This conditional parameter specifies the Type X address from which the service was requested. It may be the address of the Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device from which the service was requested. It is present if the service was requested from a Type "X" device.

#### 7.2.2.3.6.3 Service procedure

The Confirmed Service Procedure specified in Clause 4 applies to this service. The service fails if the Device ID and Device Type parameters do not match those of the addressed SMK.

### 7.2.2.3.7 Clear assignment info service

#### 7.2.2.3.7.1 Service overview

This service is used to clear a device by clearing SMK and LAN Redundancy assignment information and setting all VFDs to factory defaults. It may be used to clear the PD Tag of a device on a Type "X" link that is connected to a Type 5 device. If the SMK ID indicates that a Type "X" PD Tag is to be cleared, the SMK invokes the appropriate Type "X" SM procedures to clear it.

The configuration application includes the PD Tag (which may be blank) and Device ID to verify that the correct device will receive this service. If either the PD Tag or the Device ID do not match, the device returns a negative response.

#### 7.2.2.3.7.2 Service parameters

The service parameters for this service are shown in Table 115.

**Table 115 – Clear assignment info service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| SMK ID | M | M (=) | | |
| Device ID | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Error Info | | | M | M (=) |

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

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the service was requested.

**Destination Address**

This parameter is the address to which the request is sent.

**SMK ID**

This parameter identifies the SMK that is to respond to the request. The responder may be:

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

**Device ID**

This parameter specifies the value of the SMK attribute of the same name. This value should match that of that SMK attribute.

**Physical Device Tag**

This parameter specifies the value of the SMK attribute of the same name. This value should match that of that SMK attribute.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**SMK ID**

This parameter identifies the SMK sending the response. The responder may be:

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**SMK ID**

This parameter identifies the SMK sending the response. The responder may be:

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

#### 7.2.2.3.7.3 Service procedure

The Confirmed Service Procedure specified in Clause 4 applies to this service. The service fails if the Device ID and Device Type parameters do not match those of the addressed SMK.

### 7.2.2.3.8 Clear address service

#### 7.2.2.3.8.1 Service overview

This service is used to clear the address of a Type 5 device, a Type "X" link interface connected to a Type 5 device, or a device on a Type "X" link that is connected to a Type 5 device. After clearing the address of a Type "X" link interface connected to a Type 5 device or a device on a Type "X" link that is connected to a Type 5 device, the responding device SMK returns the response. After accepting the request to clear its own address, the device SMK returns the response, and then clears its address.

This service operates over dynamically established or pre-established client/server sessions.

#### 7.2.2.3.8.2 Service parameters

The service parameters for this service are shown in Table 116.

**Table 116 – Clear address service parameters**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Destination Address | M | M (=) | | |
| SMK ID | C | C (=) | | |
| Device ID | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| InterfaceToClear | C | C (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Destination Address | | | M | M (=) |
| SMK ID | | | M | M (=) |
| Error Info | | | M | M (=) |

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

### Argument

The argument contains the parameters of the service request.

### Destination Address

This parameter is the address to which the service request is to be sent. The request may addressed to a Type 5 device, to a Type "X" link connected to a Type 5 device, or to a device on a Type "X" link that is connected to a Type 5 device.

### SMK ID

This parameter identifies the SMK that is to respond to the request. The responder may be:

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

### Device ID

This parameter specifies the value of the SMK attribute of the same name. This value should match that of that SMK attribute.

### Physical Device Tag

This conditional parameter specifies the value of the SMK attribute of the same name. This value should match that of that SMK attribute.

**InterfaceToClear**

This parameter indicates which of the device's network interfaces are to the cleared. It is present if the request is addressed to a Type 5 SMK.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Destination Address**

This parameter is the address to which the service response is to be returned.

**SMK ID**

This parameter identifies the SMK that responds to the request. The responder may be:

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Destination Address**

This parameter is the address to which the error service response is to be returned.

**SMK ID**

This parameter identifies the SMK that responds to the request. The responder may be:

– a Type 5 SMK, or

– a Type "X" SMK of a Type "X" device connected to a Type 5 linking device.

#### 7.2.2.3.8.3 Service procedure

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

In addition, the Device ID, Physical Device Tag, and Device Type parameter values should match the corresponding attribute values in the SMK of the device that is to have its address cleared request. If they do not match, a negative response is returned.

### 7.2.3 LAN redundancy ASE

#### 7.2.3.1 Overview

Devices can have two network interfaces that are used for LAN redundancy. These interfaces are referred to as Network Interface A and Network Interface B. They are managed by a user of the LAN Redundancy ASE referred to as the LAN Redundancy Entity.

The LAN Redundancy Entity exchanges redundancy status information with other LAN Redundancy Entities using the Diagnostic Message Service. The LAN Redundancy Entity of a device selects which network interface to use to send other APDUs based on this status information.

Each LAN Redundancy Entity submits a Diagnostic Message Service request primitive at periodic intervals and the LAN Redundancy ASE constructs two Diagnostic Message Service APDUs and sends them on Network Interface A and Network Interface B concurrently. Each LAN Redundancy Entity receives pairs of service indications from the other devices on the network that participate in LAN Redundancy. This information is used to determine which network interface (A or B) is to be selected to send other APDUs.

### 7.2.3.2 LAN redundancy class specification

#### 7.2.3.2.1 LAN redundancy formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **LAN Redundancy** |
| **CLASS:** | | | **LAN Redundancy** |
| **CLASS ID:** | | | Not used |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribute: | LAN Redundancy Attributes Version |
| 2 | (m) | Attribute: | Number of Network Interfaces |
| 3 | (m) | Attribute: | Max Sequence Number Difference |
| 4 | (m) | Attribute: | LAN Redundancy Flags |
| 5 | (m) | Attribute: | Diagnostic Message Interval |
| 6 | (m) | Attribute: | Aging Time |
| 7 | (m) | Attribute: | Diagnostic Message Send Addresses |
| 8 | (m) | Attribute: | Diagnostic Message Receive Addresses |
| 9 | (m) | Attribute: | List of Network Interface Status |
| 10 | (m) | Attribute: | Number of Diagnostic Service Indications Received |
| 11 | (m) | Attribute: | Number of Diagnostic Message Indications Missed |
| 12 | (m) | Attribute: | Number of Faults Detected |
| 13 | (m) | Constraint: | Crossed Cable Detection Enabled = TRUE |
| 13.1 | (m) | Attribute: | Number of Diagnostic Service Indications Missed |
| **SERVICES:** | | | |
| 1 | (m) | OpsService: | Diagnostic Message |
| 2 | (o) | OpsService: | Get Redundancy Information |
| 3 | (o) | OpsService: | Put Redundancy Information |
| 4 | (o) | OpsService: | Get Redundancy Statistics |

#### 7.2.3.2.2 Attributes

**LAN Redundancy Attributes Version**

This attribute specifies the version of this object. Each time the value of any of its other attributes changes, the version number is incremented by 1. It survives a power-fail if the APDU contents survive the power-fail; otherwise its value is reset to 0. This is the only time that its value is reset to 0. Version number 0 indicates that this object has not been configured.

**Number of Network Interfaces**

Specifies the number of network interfaces on this device. A device may have one or two network interfaces. They are labelled Network Interface A and Network Interface B. (Network Interface B is only used if there are two network interfaces.)

**Max Sequence Number Difference**

This attribute defines the maximum acceptable difference between the sequence number parameters in a pair of Diagnostic Message Service indications received from a single sending device. When the difference exceeds the value of this attribute, a fault in the path from the network interface sending the lower sequence number is detected.

**LAN Redundancy Flags**

This attribute controls how the Diagnostic Message Service is used by the LAN Redundancy Entity. Its value may indicate one or more of the following:

– Single Multicast APDU Transmission Interface Enabled. The transmission policy for all services with multicast destination addresses except for the Diagnostic Message and SMK Annunciate services.

False Transmit on both network interfaces

True Transmit on one network interface

– Crossed Cable Detection Enabled.

False Do not detect crossed cables

True Detect crossed cables

– Single Multicast APDU Reception Interface Enabled. The reception policy for all multicast destination addresses except for SM Multicast address and the LAN Diagnostic Message Receive Addresses for interfaces A and B.

False Listen for multicast addresses on both network interfaces

True Listen for multicast addresses on one network interface if zero or one fault detected

– Diagnosis using own APDUs Enabled.

False Do not use own diagnostic messages for diagnosis

True Use own diagnostic messages for diagnosis

– Load Balancing Enabled.

False Do not do load balancing

True Do load balancing

**Diagnostic Message Interval**

This attribute specifies the time interval in milliseconds between successive invocations of the Diagnostic Message Service by the LAN Redundancy Entity.

**Aging Time**

This attribute specifies the time interval used by the LAN Redundancy Entity to remove silent devices from its list of devices that are participating in LAN redundancy. The value of this attribute should be larger than the value of the *Max Sequence Number Difference* attribute multiplied by the largest value of the parameter *Diagnostic Message Interval* found in Diagnostic Message Service indications.

**Diagnostic Message Send Addresses**

This attribute defines the destination addresses for sending Diagnostic Message APDUs. One address is defined for each network interface from which Diagnostic Message APDUs are sent. The addresses may be a broadcast, multicast or unicast addresses.

**Diagnostic Message Receive Addresses**

This attribute defines the addresses for receiving Diagnostic Message APDUs. One address is defined for each network interface on which Diagnostic Message APDUs are received. The addresses may be a broadcast, multicast or unicast addresses.

**List of Network Interface Status**

This attribute is a list. Each entry in this list represents the status for the path from a remote device from which Diagnostic Messages Service indications may be received. The status is bidirectional for path and indicates whether or not Diagnostic Messages Service indications are being successfully received at each end of the path or whether either end of the path is in a fault state (a fault has been detected and has not cleared).

**Number of Diagnostic Message Service Indications Received**

This attribute counts the number of Diagnostics Message Service indications received (from all devices). A separate count is maintained for each network interface.

**Number of Diagnostic Message Service Indications Missed**

This attribute is a count of the number of missed Diagnostics Message Service indications (from all devices) as computed from sequence number gaps. A separate count is maintained for each network interface.

**Number of Faults Detected**

This attribute is a count of the total number of faults detected from missed Diagnostics Message Service indications. A separate count is maintained for each network interface.

**List of Crossed Cable Status**

This conditional attribute is a list. This attribute is present only if the value of the *Crossed Cable Detection Enabled* component of the LAN Redundancy Flags attribute is TRUE. Each entry in this list represents the Crossed Cable Status for the path to a remote device from which Diagnostic Messages Service indications may be received. The Crossed Cable Status indicates whether the network interface used by the Diagnostic Message Service for requests, as indicated by the *Network Interface used for Transmission of this Diagnostic Message* parameter in the Diagnostic Message Service, is the same network interface used by the reporting device to receive the messages.

#### 7.2.3.2.3 Services

**Diagnostic Message**

This service is used to send the same diagnostic message concurrently from each network interface.

**Get Redundancy Information**

This service is used to retrieve LAN Redundancy attributes.

**Set Redundancy Information**

This service is used to update LAN Redundancy attributes.

**Get Redundancy Statistics**

This service is used to retrieve LAN Redundancy statistics.

### 7.2.3.3 LAN redundancy ASE service specification

#### 7.2.3.3.1 Overview

This subclause contains the definition of the services that is unique to this ASE. LAN Redundancy Services operate directly over socket services. They do not use ARs.

The services defined for this ASE are:

Diagnostic Message Service
Get Redundancy Info Service
Put Redundancy Info Service
Get Redundancy Statistics Service

#### 7.2.3.3.2 Diagnostic message service

##### 7.2.3.3.2.1 Service overview

This service is used to cause Diagnostic Message Service APDUs to be sent simultaneously from each redundant network interface, labeled Interface A and Interface B. The messages are identical except for the identification of which interface was used to send each APDU.

##### 7.2.3.3.2.2 Service parameters

The service parameters for this service are shown in Table 117.

**Table 117 – Diagnostic message service**

| Parameter name | Req | Ind |
|---|---|---|
| Argument | | |
| Source Address | | M |
| Destination Address | M | M (=) |
| Device Index | M | M (=) |
| Number of Interfaces | M | M (=) |
| Transmission Interface | M | M (=) |
| Diagnostic Message Interval | M | M (=) |
| SMK PD Tag | M | M (=) |
| SMK Duplicate Detection State | M | M (=) |
| List of Network Interface Status | M | M (=) |

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the request was sent.

**Destination Address**

This parameter is the address to which the request was sent.

**Device Index**

This value is obtained from the SMK attribute of the same name.

**Number of Network Interfaces**

This value is derived from the SMK List of Network Addresses attribute. There is one network address per network interface in this list.

**Transmission Interface**

This parameter is the transmission interface from which the request was sent.

**Diagnostic Message Interval**

This value is obtained from the LAN Redundancy attribute of the same name.

**SMK PD Tag**

This parameter specifies the value of the SMK PD Tag attribute of the device.

**SMK Duplicate Detection State**

This parameter specifies the value of the SMK Duplicate Detection State attribute of the device.

**List of Network Interface Status**

This value is obtained from the LAN Redundancy attribute of the same name.

#### 7.2.3.3.2.3 Service procedure

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

### 7.2.3.3.3 Get redundancy info service

#### 7.2.3.3.3.1 Service overview

This confirmed service is used to retrieve LAN Redundancy attributes.

#### 7.2.3.3.3.2 Service parameters

The service parameters for this service are shown in Table 118.

**Table 118 – Get redundancy info service**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| | | | | |
| LAN Redundancy Attributes Version | | | M | M (=) |
| Number of Network Interfaces | | | M | M (=) |
| Max Sequence Number Difference | | | M | M (=) |
| LAN Redundancy Flags | | | M | M (=) |
| Diagnostic Message Interval | | | M | M (=) |
| AgingTime | | | M | M (=) |
| Diagnostic Message Interface Send Addresses | | | M | M (=) |
| Diagnostic Message Interface Receive Addresses | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | M | M (=) |

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

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the request was sent.

**Destination Address**

This parameter is the address to which the request was sent.

**LAN Redundancy Attributes Version**

This value contains the value of the LAN Redundancy attribute of the same name.

**Number of Network Interfaces**

This value contains the value of the LAN Redundancy attribute of the same name.

**Max Sequence Number Difference**

This value contains the value of the LAN Redundancy attribute of the same name.

**LAN Redundancy Flags**

This value contains the value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interval**

This value contains the value of the LAN Redundancy attribute of the same name.

**AgingTime**

This value contains the value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interface Send Addresses**

This value contains the value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interface Receive Addresses**

This value contains the value of the LAN Redundancy attribute of the same name.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

#### 7.2.3.3.3.3 Service procedure

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

### 7.2.3.3.4 Put redundancy info service

#### 7.2.3.3.4.1 Service overview

This confirmed service is used to retrieve LAN Redundancy attributes. After updating the attributes contained in the request, the responder returns the values and the updated LAN Redundancy Attributes Version number.

#### 7.2.3.3.4.2 Service primitives

The service parameters for this service are shown in Table 119.

**Table 119 – Put redundancy info service**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| | | | | |
| LAN Redundancy Attributes Version | | | M | M (=) |
| Number of Network Interfaces | | | M | M (=) |
| Max Sequence Number Difference | | | M | M (=) |
| LAN Redundancy Flags | | | M | M (=) |
| Diagnostic Message Interval | | | M | M (=) |
| AgingTime | | | M | M (=) |
| Diagnostic Message Interface Send Addresses | | | M | M (=) |
| Diagnostic Message Interface Receive Addresses | | | M | M (=) |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| | | | | |
| LAN Redundancy Attributes Version | | | M | M (=) |
| Number of Network Interfaces | | | M | M (=) |
| Max Sequence Number Difference | | | M | M (=) |
| LAN Redundancy Flags | | | M | M (=) |
| Diagnostic Message Interval | | | M | M (=) |
| AgingTime | | | M | M (=) |
| Diagnostic Message Interface Send Addresses | | | M | M (=) |
| Diagnostic Message Interface Receive Addresses | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | M | M (=) |

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

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the request was sent.

**Destination Address**

This parameter is the address to which the request was sent.

**LAN Redundancy Attributes Version**

This value contains the value of the LAN Redundancy attribute of the same name.

**Number of Network Interfaces**

This value contains the new value of the LAN Redundancy attribute of the same name.

**Max Sequence Number Difference**

This value contains the new value of the LAN Redundancy attribute of the same name.

**LAN Redundancy Flags**

This value contains the new value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interval**

This value contains the new value of the LAN Redundancy attribute of the same name.

**AgingTime**

This value contains the new value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interface Send Addresses**

This value contains the new value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interface Receive Addresses**

This value contains the new value of the LAN Redundancy attribute of the same name.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**LAN Redundancy Attributes Version**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**Number of Network Interfaces**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**Max Sequence Number Difference**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**LAN Redundancy Flags**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interval**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**AgingTime**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interface Send Addresses**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**Diagnostic Message Interface Receive Addresses**

This value contains the updated value of the LAN Redundancy attribute of the same name.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

#### 7.2.3.3.4.3 Service procedure

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

### 7.2.3.3.5 Get redundancy statistics service

#### 7.2.3.3.5.1 Service overview

This confirmed service is used to retrieve LAN Redundancy statistics attributes.

#### 7.2.3.3.5.2 Service primitives

The service parameters for this service are shown in Table 120.

**Table 120 – Get redundancy statistics service**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Number of Diagnostic Message Service Indications Received | | | M | M (=) |
| Number of Diagnostic Message Service Indications Missed | | | M | M (=) |
| Number of Faults Detected | | | M | M (=) |
| List of Crossed Cable Status | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | M | M (=) |

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

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the request was sent.

**Destination Address**

This parameter is the address to which the request was sent.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**Number of Diagnostic Message Service Indications Received**

This value contains the value of the LAN Redundancy attribute of the same name.

**Number of Diagnostic Message Service Indications Missed**

This value contains the value of the LAN Redundancy attribute of the same name.

**Number of Faults Detected**

This value contains the value of the LAN Redundancy attribute of the same name.

**List of Crossed Cable Status**

This value contains the value of the LAN Redundancy attribute of the same name.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

#### 7.2.3.3.5.3 Service procedure

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

## 7.3 FDA sessions

### 7.3.1 Publisher/subscriber session endpoint class specification

#### 7.3.1.1 Class overview

This class is defined to support the on-demand queued distribution of unconfirmed services to one or more application processes. It is the Type 5 counterpart to the Type 9 BNU AREPs. The behaviour of this type of AR can be described as follows.

An AR ASE user wishing to convey an unconfirmed APDU submits an AR ASE Service Data Unit to the sending endpoint of the AR. The AREP sending the request APDU submits it to the underlying layer for transfer. The underlying layer sends it at its next opportunity. The AREP receiving the request APDU from its underlying layer delivers it to selected AR ASE users in the order that it was received. The AR ASE users are selected based on the address information contained in the AR header.

The sending endpoint has a role of PUBLISHER and the receiving endpoints have the role of SUBSCRIBER.

The following summarises the characteristics of this AREP class:

| | |
|---|---|
| Roles: | PUBLISHER<br>SUBSCRIBER |
| Cardinality: | 1-to-n |
| Timeliness: | No |

#### 7.3.1.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | | | **Publisher/Subscriber Session Endpoint** |
| **CLASS ID:** | | | **Not Used** |
| **PARENT CLASS:** | | | **TOP** |
| **NETWORK MANAGEMENT ATTRIBUTES:** | | | |
| 1. | (m) | Attribute: | Role (PUBLISHER, SUBSCRIBER) |
| 2. | (m) | Attribute: | AREP State (OPEN, CLOSED) |
| 3. | (m) | Attribute: | Socket Mapping Reference |
| **SERVICES:** | | | |
| 1. | (o) | OpsService: | Unconfirmed Send |

#### 7.3.1.3 Network management attributes

**Role**

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

PUBLISHER Endpoints of this type send messages to multiple subscriber endpoints using unconfirmed services operating over connectionless socket services.

SUBSCRIBER Endpoints of this type receive reports from source endpoints using unconfirmed services operating over connectionless socket services.

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Socket Mapping Reference**

This attribute is a reference to the socket mapping to the underlying layer. These mappings are specified in IEC 61158-6-5.

#### 7.3.1.4 Services

**Unconfirmed Send**

This optional service is used to send an unconfirmed service on an AR. This service is specified by Type 9.

### 7.3.2 Report distribution session endpoint class specification

#### 7.3.2.1 Class overview

This class is defined to support the on-demand queued distribution of unconfirmed services to one or more application processes. It is the Type 5 counterpart to the Type 9 QUU AREPs. The behaviour of this type of AR can be described as follows.

An AR ASE user wishing to convey an unconfirmed APDU submits an AR ASE Service Data Unit to the sending endpoint of the AR. The AREP sending the request APDU submits it to the

underlying layer for transfer. The underlying layer sends it at its next opportunity. The AREP receiving the request APDU from its underlying layer delivers it to all AR ASE users of the receiving AREP in the order that it was received.

The sending endpoint has a role of REPORT SOURCE and the receiving endpoints have the role of REPORT SINK.

The following summarises the characteristics of this AREP class:

| | |
|---|---|
| Roles: | REPORT SOURCE<br>REPORT SINK |
| Cardinality: | 1-to-n |
| Timeliness: | No |

#### 7.3.2.2 Formal model

**FAL ASE:** **AR ASE**
**CLASS:** **Report Distribution Session Endpoint**
**CLASS ID:** **Not Used**
**PARENT CLASS:** **TOP**
**NETWORK MANAGEMENT ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1. | (m) | Attribute: | Role (REPORT SOURCE, REPORT SINK) |
| 2. | (m) | Attribute: | AREP State (OPEN, CLOSED) |
| 3. | (m) | Attribute: | Socket Mapping Reference |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1. | (o) | OpsService: | Unconfirmed Send |

#### 7.3.2.3 Network management attributes

**Role**

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

REPORT SOURCE endpoints of this type send messages to multiple REPORT SINK endpoints using unconfirmed services operating over connectionless socket services.

REPORT SINK endpoints of this type receive reports from REPORT SOURCE endpoints using unconfirmed services operating over either connectionless.

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Socket Mapping Reference**

This attribute is a reference to the socket mapping to the underlying layer. These mappings are specified in IEC 61158-6-5.

#### 7.3.2.4 Services

**Unconfirmed Send**

This optional service is used to send an unconfirmed service on an AR. This service is specified by Type 9.

### 7.3.3 Client/server AR endpoint class specification

#### 7.3.3.1 Class overview

This class is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. It is the Type 5 counterpart to the Type 9 QUB Co AREP. The behaviour of this type of AR can be described as follows.

An AR ASE user wishing to convey a request or response APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity. Either endpoint may send unconfirmed request or response APDUs. Only the Client AREP may send the OpenSession Request APDU.

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

The following summarises the characteristics of this AREP class:

| | |
|---|---|
| Roles: | Client<br>Server |
| Cardinality: | 1-to-1 |
| Timeliness: | No |

#### 7.3.3.2 Formal model

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **AR ASE** |
| **CLASS:** | | | **Client/server AREP** |
| **CLASS ID:** | | | **Not Used** |
| **PARENT CLASS:** | | | **TOP** |
| **NETWORK MANAGEMENT ATTRIBUTES:** | | | |
| 1. | (m) | Attribute: | Role (CLIENT, SERVER) |
| 2. | (m) | Attribute: | AREP State |
| 3. | (m) | Attribute: | Server Physical Device Tag |
| 4. | (m) | Attribute: | Socket Mapping Reference |
| **SERVICES:** | | | |
| 1. | (o) | OpsService: | Confirmed Send |
| 2. | (o) | OpsService: | Unconfirmed Send |
| 3. | (o) | OpsService: | OpenSession |

#### 7.3.3.3 Network management attributes

**Role**

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

CLIENT Endpoints of this type issue confirmed service Request-APDUs to initiate the establishment of the AR. One and only one of the AREPs involved in an AR should be the initiator.

SERVER Endpoints of this type respond to requests received from the CLIENT to establish the AR.

**AREP State**

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6-5.

**Server Physical Device Tag**

This conditional attribute specifies the Physical Device Tag SMK attribute of the server device. This attribute is used to locate the server on the network. How the server is located is beyond the scope of this standard. This attribute is present only for CLIENT AREPs.

**Socket Mapping Reference**

This attribute is a reference to the socket mapping to the underlying layer. These mappings are specified in IEC 61158-6-5.

#### 7.3.3.4 Services

**Unconfirmed Send**

This optional service is used to send an unconfirmed service on an AR. This service is specified by Type 9.

**Confirmed Send**

This optional service is used to send a confirmed service on an AR. This service is specified by Type 9.

**OpenSession**

This service is used to establish an AR.

### 7.3.4 FDA session ASE service specification

#### 7.3.4.1 Supported services

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

The services defined for this ASE are:

Open Session Service
Idle Service

#### 7.3.4.2 Open session service

##### 7.3.4.2.1 Overview of how the service might be implemented in a protocol

NOTE The following description presumes a protocol such as that of IEC 61158-6-5 that implements these services. Other protocols that implement these services are possible.

This service is used to open a Client/Server session between a client session endpoint and a server session endpoint. The source and destination network address pair identifies each session endpoint, where each address is composed of the network device address and a device internal selector.

Client session endpoints are configured with the PD Tag of the server instead of with the network address of the server. They may use the SM Find Tag Query to determine the destination network address or they may obtain it from the Periodic Annunciations sent by the server.

Use of the Find Tag Query service may be necessary also when reopening a session after failure. In this case, the cause of the failure may be failure of the server. If the server is redundant, the secondary will take over, but it will use its own network address. Therefore, the client needs to determine if the network address associated with the server's PD Tag has changed.

Once the network address of the server has been determined, the client sends the FDA Open Session request message to the FDA Selector located at the network device address.

The Version field in the FDA Message Header in the request message indicates the version of the FDA that is being requested for the session. The responder may negotiate the version down.

The Options field in the Message Header in the request message indicates the options that are being requested for the session. The Instance Id bit is always set. The responder can refuse all other options. Returning 0 in the bit position representing the option indicates refusal.

If the message number option is requested, then the trailer field contains the initial value to be used by both endpoints of the session. This value is returned in the response message. If the requester does not receive a response to the request, and elects to resend the request, it increments the message number in the trailer, while keeping the invoke id the same as the original. This causes the request to be delivered to the session endpoint created if the first request was received and a positive response was returned. If the original request was not received, or it caused a negative response to be returned (that was not received), then the new request is treated like a new request because there is no existing session endpoint to process the message.

The FDA Address in the Message Header is not used and is set to 0.

Successful exchange of Open Session request and response messages results in the establishment of a session that may be used to send request and response messages. If connectionless services are used, the server returns the response from a previously unused connectionless selector. This port is then used to send and receive all subsequent messages for the session.

If the responder is does not have an available Session Index for the new session, then a negative response is returned with error class = “resource” and error code = “max sessions exceeded”.

If the responder is does not have the resources to support the new session, then a negative response is returned with error class = “resource” and error code = “object creation failure” or some other appropriate error code within the “resource” error class.

The Open Session request message conveys session attributes for validation and use by the responder. If the responder is able to support certain of these, it is permitted to negotiate them, as defined in the Request Message Parameters in Table 121 below. If the requester is not prepared to operate with the negotiated attributes returned by the responder, it is free to not open the session. It may then reissue an Open Session request with different session attributes, or if using connection-oriented services, close the associated connection to close the session, as appropriate.

If a request is received with invalid or unsupported parameter values, a negative response is returned with an error class of "service" and error code of "parameter-inconsistent" for the offending parameter. This negative response also uses the additional code value of 1, and the additional description to propose an acceptable values for each of the parameters indicated below that can be used to open the session. To convey these values the 16 octets of the additional description is interpreted as 4 Unsigned32 integers instead of as a VisibleString. The location of each parameter value in the 16 octets is specified in the descriptions below.

If the responder receives an Open Session request message for a session on which it has already sent an Open Session response message, it closes the session.

If the responder is does not have an available Session Index for the new session, then a negative response is returned with error class = “resource” and error code = “max sessions exceeded”.

If the responder is does not have the resources to support the new session, then a negative response is returned with error class = “resource” and error code = “object creation failure” or some other appropriate error code within the “resource” error class.

Sessions opened for NMA Configuration Use are referred to as configuration sessions. Only one configuration session is allowed at a time. If a request is received to open a configuration session and one is already open, or, optionally, if there is VCR to any Type 5 or Type 9 MIB with update services supported already open, then the configuration session is refused. The FDA Agent returns a negative response with error class = "access" and error code = "config-access-already-open".

Care needs to be taken when negotiating the inactivity time period for configuration sessions that use connectionless services. If the client fails, the server is not permitted to allow another configuration client to establish a session until the negotiated inactivity time period expires.

Opening publisher/subscriber and Report Distribution sessions is local, and does not result in the exchange of FDA Messages. As a result, the Open Session message parameters for these session types are configured into the NMIB instead of negotiated.

#### 7.3.4.2.2 Service primitives

The service parameters for this service are shown in Table 121.

**Table 121 – Open session service**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| Session Index | M | M (=) | | |
| Max Buffer Size | M | M (=) | | |
| Max Message Length | M | M (=) | | |
| NMA Configuration Use | M | M (=) | | |
| Inactivity Close Time | M | M (=) | | |
| Transmit Delay Time | M | M (=) | | |
| PD Tag | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Session Index | | | M | M (=) |
| Max Buffer Size | | | M | M (=) |
| Max Message Length | | | M | M (=) |
| NMA Configuration Use | | | M | M (=) |
| Inactivity Close Time | | | M | M (=) |
| Transmit Delay Time | | | M | M (=) |
| PD Tag | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | M | M (=) |

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

**Argument**

The argument contains the parameters of the service request.

**Source Address**

This parameter is the address from which the request was sent.

**Destination Address**

This parameter is the address to which the request was sent.

**Session Index**

The Session Index is the OD Index of the session endpoint description of the client session endpoint, if one exists. If not, 0 is used.

**Max Buffer Size**

This parameter defines the maximum buffer length in octets (the buffer may contain concatenated FDA Messages) that the sender of this message may send or receive on this session. This parameter can be negotiated down (but not up) by the responder using the Max Buffer Size parameter. If the negotiation fails for any parameter, an acceptable value for this parameter is returned in the additional description field as an Unsigned32 at offset 0. If the requested value is supported, but another parameter failed the negotiation, then the requested value is returned at offset 0.

**Max Message Length**

This parameter defines the maximum length in octets of messages to be sent on this session by the sender of this message. It is used by the receiving session endpoint as its Max Message Length attribute. If the negotiation fails for any parameter, an acceptable value for this parameter is returned in the additional description field as an Unsigned32 at offset 4. If the requested value is supported, but another parameter failed the negotiation, then the requested value is returned at offset 4.

**NMA Configuration Use**
0 = NMA Configuration Not Permitted

1 = NMA Configuration Permitted

This parameter cannot be negotiated.

**Inactivity Close Time**

This parameter identifies how long, in seconds, that the session stays open without receiving a message. After experiencing inactivity on the session for this period of time, the session endpoint is closed, along with all VCR activity associated with the session endpoint. The responder can negotiate this value down. The value 0 is not permitted. If the negotiation fails for any parameter, an acceptable value for this parameter is returned in the additional description field as an Unsigned32 at offset 8. If the requested value is supported, but another parameter failed the negotiation, then the requested value is returned at offset 8.

**Transmit Delay Time**

This parameter is used to set the Transmit Delay Time attribute in the receiving session endpoint. Its value may not be negotiated. Its value is expressed in milliseconds. If the negotiation fails for any parameter, an acceptable value for this parameter is returned in the additional description field as an Unsigned32 at offset 12. If the requested value is supported, but another parameter failed the negotiation, then the requested value is returned at offset 12.

**PD Tag**

PD Tag of the server. If the PD Tag in the request message does not match the PD Tag of the server, then the server rejects the request with error class “access” and error code “object-access-denied”.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

**Session Index**

The Session Index is the OD Index of the newly opened session.

**Max Buffer Size**

This parameter defines the negotiated buffer length in octets (the buffer may contain concatenated FDA Messages) that the sender of this message may send or receive on this session.

**Max Message Length**

This parameter defines the maximum length in octets of messages to be sent on this session by the sender of this message. It is used by the receiving session endpoint as its Max Message Length attribute.

**NMA Configuration Use**
0 = NMA Configuration Not Permitted

1 = NMA Configuration Permitted

**Inactivity Close Time**

This parameter identifies how long, in seconds, that the session stays open without receiving a message. After experiencing inactivity on the session for this period of time, the session endpoint is closed, along with all VCR activity associated with the session endpoint. The responder can negotiate down this value. The value 0 is not permitted.

**Transmit Delay Time**

This parameter is used to set the Transmit Delay Time attribute in the receiving session endpoint. Its value is expressed in milliseconds.

**PD Tag**

PD Tag of the server.

**Result(–)**

This selection type parameter indicates that the service request failed.

**Source Address**

This parameter is the address from which the response was sent.

**Destination Address**

This parameter is the address to which the response is sent.

#### 7.3.4.3 Idle service

##### 7.3.4.3.1 Service overview

This service is used by client/server VCR endpoints to keep the VCR alive. Receipt of an Idle message informs the receiver that the sender is present. It may be sent using connectionless services or connection-oriented services. The FDA Address contains the OD Index of the dynamically established server VCR endpoint. If this endpoint does not exist, then an error message is returned with error class = "access" and error code = "unrecognized FDA Address".

##### 7.3.4.3.2 Service primitives

The service parameters for this service are shown in Table 122.

**Table 122 – Idle session service**

| Parameter name | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| VCR Id | | M | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| VCR Id | | | | M |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| VCR Id | | | | M |
| Error Info | | | M | M (=) |

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

**Argument**

The argument contains the parameters of the service request.

**VCR Id**

This parameter identifies the VCR of the service.

**Result(+)**

This selection type parameter indicates that the service request succeeded.

**VCR Id**

This parameter identifies the VCR of the service.

**Result(–)**

This selection type parameter indicates that the service request failed.

**VCR Id**

This parameter identifies the VCR of the service.

### 7.4 Summary of FAL Type 9 and Type 5 classes

Table 123 contains a summary of the defined FAL Classes. The Class ID values are defined in the corresponding subclauses of Clause 15 (Type 9).

**Table 123 – FAL class summary**

| FAL ASE | Class | Class ID |
|---|---|---|
| VFD | VFD | — |
| | VCR | |
| Data type | Fixed Length & String Data type | See Type 9 |
| | Structure Data type | See Type 9 |
| Object Dictionary | OD Description | — |
| | Object Dictionary | — |
| Type 5 AR (FDA Session) | Client/server | — |
| | Publisher/subscriber | — |
| | Report Distribution | — |
| Type 9 Variable | Simple Variable | See Type 9 |
| | Array Variable | See Type 9 |
| | Record Variable | See Type 9 |
| | Variable List | See Type 9 |
| Type 9 Event | Event | See Type 9 |
| Type 9 Load Region | Load Region | See Type 9 |
| Type 9 Function Invocation | Function Invocation | See Type 9 |
| SMK | SMK | — |
| LAN Redundancy | LAN Redundancy | — |

## 7.5 Permitted FAL Type 9 and Type 5 services by AREP role

Table 124 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 124 – Services by AREP role**

| | Unconfirmed | Confirmed | Client | | Server | | Publish | Subscribe | |
|---|---|---|---|---|---|---|---|---|---|
| **FAL Services** | | | **req** | **rcv** | **req** | **rcv** | **req rcv** | **req** | **rcv** |
| **Type 9 Mgt ASE** | | | | | | | | | |
| Create | | X | X | X | X | X | | | |
| Delete | | X | X | X | X | X | | | |
| Get Attributes | | X | X | X | X | X | | | |
| Get Attribute List | | X | X | X | X | X | | | |
| Set Attributes | | X | X | X | X | X | | | |
| Begin Set Attributes | | X | X | X | X | X | | | |
| End Set Attributes | | X | X | X | X | X | | | |
| **VFD ASE** | | | | | | | | | |
| Identify | | X | X | X | X | X | | | |
| Get Status | | X | X | X | X | X | | | |
| Status Notification | X | | X | X | X | X | X | | X |
| Initiate | | X | X | X | X | X | | | |
| Terminate | | X | X | X | X | X | | | |
| **FDA Session ASE** | | | | | | | | | |
| Confirmed Send | | | X | X | X | X | | | |
| Unconfirmed Send | | | X | X | X | X | X | | X |
| Open Session | | | X | | | X | | | |
| **Type 9 AR ASE** | | | | | | | | | |
| AR-Confirmed Send | | | X | X | X | X | | | |
| AR-Unconfirmed Send | | | X | X | X | X | X | | X |
| AR-Associate | | | X | | | X | | | |
| AR-Abort | | | X | X | X | X | X | X | X |
| **Type 9 Variable ASE** | | | | | | | | | |
| Read | | X | X | X | X | X | | | |
| Write | | X | X | X | X | X | | | |
| Information Report | X | | X | X | X | X | X | | X |
| **Type 9 Event ASE** | | | | | | | | | |
| Confirmed Ack Event | | X | X | X | X | X | | | |
| Enable Event | | X | X | X | X | X | | | |
| Event Notification | X | | X | X | X | X | X | | X |
| **Type 9 Domain ASE** | | | | | | | | | |
| Initiate Load | | X | X | X | X | X | | | |
| Push Segment | | X | X | X | X | X | | | |
| Pull Segment | | X | X | X | X | X | | | |
| Terminate Load | | X | X | X | X | X | | | |

| FAL Services | Unconfirmed | Confirmed | Client | | Server | | Publish | | Subscribe | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | req | rcv | req | rcv | req | rcv | req | rcv |
| **Type 9 Program Invocation ASE** | | | | | | | | | | |
| Start | | X | X | X | X | X | | | | |
| Stop | | X | X | X | X | X | | | | |
| Resume | | X | X | X | X | X | | | | |
| Reset | | X | X | X | X | X | | | | |
| Kill | | | X | X | X | X | | | | |
| **LAN Redundancy ASE** | | | | | | | | | | |
| Diagnostic Message | X | | | | | | X | | X | |
| **SMK ASE** | | | | | | | | | | |
| Find Tag Query | X | | X | X | X | X | X | | X | |
| Find Tag Reply | X | | X | X | X | X | X | | X | |
| Identify | | X | X | X | X | X | | | | |
| Annunciate | X | | | | | | X | | X | |
| Device Assignment | | X | X | X | X | X | X | | X | |
| Clear Address | | X | X | X | X | X | X | | X | |

# Bibliography

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 5-5: Définition des services de la couche application – Éléments de type 5

## 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 attepoint d'extrémité 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 61158-5-5 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 seconde édition annule et remplace la première édition, parue en 2007. Cette édition constitue une révision technique. La principale modification par rapport à l'édition précédente est énumérée ci-dessous:

- Bourrage de message ajouté

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*, peut être consultée sur le site web de la CEI.

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

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

**IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.**

# INTRODUCTION

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

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

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

# RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

# Partie 5-5: Définition des services de la couche application – Éléments de type 5

## 1 Domaine d'application

### 1.1 Généralités

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

La présente norme fournit les 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'automatisation et le matériau spécifique au bus de terrain de Type 5. Le terme "à temps critique" indique la présence d'une fenêtre temporelle, dans les limites de laquelle une ou plusieurs actions spécifiées sont tenues d'être parachevées avec un certain niveau de certitude défini. L'impossibilité de parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent ces actions, avec un risque concomitant pour l'équipement, l'installation et éventuellement pour la vie humaine.

La présente norme définit de manière abstraite le service visible de l'extérieur fourni par la couche application de bus de terrain de Type 5 en termes

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

b) des actions et événements primitifs du service;

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

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

Le but de la présente norme est de 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 au niveau de la frontière entre la couche application et la Gestion des systèmes selon le modèle de référence de bus de terrain.

La présente norme spécifie la structure et les services de la couche application des bus de terrain de Type 2, en conformité avec le Modèle de référence de base de l'OSI (ISO/CEI 7498) et la structure de la couche application de l'OSI (ISO/CEI 9545).

Les services et protocoles de la FAL sont fournis par des entités d'application (application entity, AE) de la FAL contenues dans les processus d'application. L'AE de la FAL se compose d'un jeu d'éléments de service application (ASE, Application Service Element) orientés objet et d'une entité de gestion de couche (LME, Layer Management Entity) qui gère l'AE. Les ASE fournissent des services de communication qui fonctionnent sur un jeu de classes d'objets de processus application (APO, application process object) connexes. L'un des ASE de la FAL est un ASE de gestion qui fournit un jeu commun de services pour la gestion des instances de classes de la FAL.

Bien que ces services spécifient, du point de vue des applications, la manière dont la demande et les réponses sont émises et distribuées, ils n'incluent pas la spécification de ce qu'il faut que les applications qui demandent et qui répondent en fassent. À 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 de support sont également définis dans la présente norme pour fournir l'accès à la FAL afin de maîtriser certains aspects de son fonctionnement.

### 1.2 Spécifications

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

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles de communications industrielles préexistants. C'est ce dernier objectif qui donne naissance à la diversité des services normalisés tels que 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 les 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 des primitives appariées "request-confirm" (c'est-à-dire demande et confirmation) ou "indication-response" (indication et réponse).

### 1.3 Conformité

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

Il n'y a pas de conformité d'équipement à la présente norme définissant des services de couche application. Au contraire, la conformité est obtenue par une mise en œuvre de protocoles de couche application conformes qui satisfont aux services de couche application de type 5 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 61131-3, *Automates programmables – Partie 3: Langages de programmation*

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*

CEI 61158-3-1, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-1: Définition des services de la couche liaison de données – Éléments de type 1*

CEI 61158-4-1, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-1: Spécification du protocole de la couche liaison de données – Éléments de type 1*

CEI 61158-5:2014 (toutes les parties), *Réseaux de communication industriels – spécifications des bus de terrain – Partie 5-5: Définition des services de la couche application*

CEI 61158-6-5, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-5: Spécification du protocole de la couche application – Éléments de type 5*

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 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de la notation de syntaxe abstraite numéro 1 (ASN.1)*[1]

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*

ANSI/IEEE 754-1985, *Binary Floating-Point Arithmetic* (disponible en anglais seulement)

## 3 Termes et définitions

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

### 3.1 Termes de l'ISO/CEI 7498-1

a) entité d'application

b) processus d'application

c) unité de données de protocole application

d) élément de service application

e) invocation d'entité d'application

f) invocation de processus d'application

g) transaction d'application

h) système ouvert réel

i) syntaxe de transfert

### 3.2 Termes de l'ISO/CEI 8822

a) syntaxe abstraite

[1] Retirée.

b) contexte de présentation

### 3.3 Termes de l'ISO/CEI 9545

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

### 3.4 Termes de l'ISO/CEI 8824

a) identificateur d'objet
b) type

### 3.5 Termes de la couche liaison de données de bus de terrain

Pour le besoins du présent document, les termes suivants tels que définis dans la CEI 61158-3-1 et dans la CEI 61158-4-1 s'appliquent.

a) DL-Time (temps de liaison de données)
b) DL-Scheduling-policy (politique de programmation de liaison de données)
c) DLCEP
d) DLC
e) DL-connection-oriented mode (mode orienté connexion de liaison de données)
f) DLPDU
g) DLSDU
h) DLSAP
i) fixed tag (marqueur fixe)
j) generic tag (marqueur générique)
k) link (liaison)
l) MAC ID (identificateur MAC)
m) network address (adresse réseau)
n) node address (adresse de nœud)
o) node (nœud)
p) tag (marqueur)
q) scheduled (programmé)
r) unscheduled (non programmé)

### 3.6 Couche application des bus de terrain – termes et définitions spécifiques

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

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

**3.6.2**
**active connection control object (Objet actif de commande de connexion)**
instance d'une certaine classe de la FAL qui identifie le moyen d'interconnexion (comme abonné et fournisseur) d'un appareil d'automation

**3.6.3**
**address assignment table (Table d'affectation d'adresses)**
mapping du stockage d'objets de données E/S internes du client avec les objets de données d'entrée et de sortie décentralisées

**3.6.4**
**allouer**
prendre une ressource dans une zone commune et affecter la ressource en question à l'usage exclusif d'une entité spécifique

**3.6.5**
**application**
fonction ou structure de données pour laquelle les données sont consommées ou produites

**3.6.6**
**interopérabilité de couche application**
capacité des entités d'application d'accomplir des opérations coordonnées et coopératives en utilisant les services de la FAL

**3.6.7**
**objets d'application**
classes d'objets multiples qui gèrent et assurent l'échange de messages pendant le mode exécution à travers le réseau et à l'intérieur de l'appareil de réseau

**3.6.8**
**processus d'application**
partie d'une application distribuée sur un réseau, qui est située sur un appareil et adressée sans ambiguïté

**3.6.9**
**identificateur de processus d'application**
composant qui distingue de multiples processus d'application utilisés dans un appareil

**3.6.10**
**objet de processus d'application**
composant d'un processus d'application qui est identifiable et accessible par l'intermédiaire d'une relation entre applications de la FAL

Note 1 à l'article: Les définitions d'objets de processus d'application se composent d'un jeu de valeurs pour les attributs de leur classe (voir la définition de "classe d'objets de processus d'application"). Les définitions d'objet de processus d'application sont accessibles à distance à l'aide des services de l'ASE Gestion d'objet de la FAL. Les services de Gestion d'objet de la FAL peuvent être utilisés pour charger ou mettre à jour des définitions d'objet, pour lire des définitions d'objet, et pour créer et supprimer de manière dynamique des objets d'application et leurs définitions correspondantes.

**3.6.11**
**classe d'objets de processus d'application**
classe d'objets de processus d'application définie en termes du jeu de leurs attributs et services accessibles par le réseau

**3.6.12**
**relation entre applications**
association coopérative entre deux ou plusieurs invocations d'entités d'application (application-entity-invocation) pour des besoins d'échange d'informations et de coordination de leur opération conjointe

Note 1 à l'article: Cette relation est activée soit par l'échange d'unités de données de protocole d'application, soit comme résultats des activités de préconfiguration.

**3.6.13**
**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 entre applications

**3.6.14**
**point d'extrémité des relations entre applications**
contexte et comportement d'une relation entre applications tels que vus et maintenus par l'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 des relations entre applications.

**3.6.15**
**attribut**
description d'une caractéristique visible de l'extérieur d'un objet

Note 1 à l'article: Les attributs d'un objet contiennent de l'information relative à des parties variables d'un objet. Typiquement, ils fournissent des informations relatives au statut ou régissent le fonctionnement d'un objet. Des attributs peuvent aussi avoir une incidence sur le comportement d'un objet. Les attributs se répartissent en attributs de classes et attributs d'instances.

**3.6.16**
**comportement**
indication de la façon dont un objet réagit à des événements particuliers

**3.6.17**
**numéro de bit**
désigne le numéro d'un bit dans une chaîne de bits (bitstring) ou dans un octet

**3.6.18**
**voie**
simple liaison physique ou logique d'un objet d'application d'entrée ou de sortie d'un serveur au processus

**3.6.19**
**diagnostic lié à une voie**
informations concernant un élément spécifique d'un objet d'application d'entrée ou de sortie, fournies pour des besoins de maintenance

EXEMPLE: validité de données

**3.6.20**
**classe**
ensemble d'objets, qui représentent tous le même type de composant système

Note 1 à l'article: Une classe est une généralisation d'un objet; un modèle pour définir des variables et des méthodes. Tous les objets dans une classe ont une forme et un comportement identiques, mais contiennent en général des données différentes dans leurs attributs.

**3.6.21**
**attribut de classe**
attribut qui est partagé par tous les objets au sein de la même classe

**3.6.22**
**code de classe**
identificateur unique attribué à chaque classe d'objets

**3.6.23**
**service spécifique à une classe**
service défini par une classe d'objets particulière pour accomplir une fonction requise qui n'est pas accomplie par un service commun

Note 1 à l'article: Un objet spécifique à une classe est unique pour la classe d'objets qui le définit.

**3.6.24**
**client**
a) objet qui utilise les services d'un autre objet (serveur) pour accomplir une tâche

b) initiateur d'un message auquel un serveur réagit

**3.6.25**
**vérification de configuration**
comparaison de la structuration attendue de l'objet de données E/S côté client avec la structuration effective de l'objet de données E/S côté serveur dans la phase de démarrage

**3.6.26**
**base de données de configuration**
ensemble d'informations d'interconnexion maintenues par l'ASE "ACCO"

**3.6.27**
**défaut de configuration**
différence inacceptable entre la structuration attendue de l'objet de données E/S et la structuration effective de l'objet de données E/S, telle que détectée par le serveur

**3.6.28**
**identificateur de configuration**
représentation d'une partie des données E/S d'un seul module d'entrée et/ou de sortie d'un serveur

**3.6.29**
**connexion**
liaison logique entre deux objets d'application qui peuvent se trouver au sein du même appareil ou dans des appareils différents

Note 1 à l'article: Les connexions peuvent être soit point à point, soit multipoint.

**3.6.30**
**voie de connexion**
description d'une connexion entre un puits et une source d'éléments de données

**3.6.31**
**ID de connexion**
**CID** (connection ID)
identificateur affecté à une émission qui est associé à une connexion particulière entre éditeurs et abonnés, donnant un nom à un élément spécifique d'informations d'application

**3.6.32**
**chemin de connexion**
train d'octets qui définit l'objet d'application auquel une instance de connexion s'applique

**3.6.33**
**point de connexion**
buffer qui est représenté comme étant une sous-instance d'un objet Assembly (ensemble)

**3.6.34**
**consommer**
recevoir des données d'un producteur

**3.6.35**
**consommateur**
nœud ou puits qui reçoit des données d'un producteur

**3.6.36**
**application consommatrice**
application qui consomme des données

**3.6.37**
**commandes de contrôle**
appels à action transférés d'un client à un serveur pour effacer les sorties, geler les entrées et/ou synchroniser les sorties

**3.6.38**
**trajet d'acheminement**
flux unidirectionnel d'unités APDU à travers une relation entre applications

**3.6.39**
**cyclique**
répétitif d'une manière régulière

**3.6.40**
**cohérence de données**
moyen pour une émission et un accès cohérents de l'objet de donnée d'entrée ou de sortie entre client et serveur et au sein du client et du serveur

**3.6.41**
**AR dédiée**
AR utilisée directement par l'utilisateur de la FAL

Note 1 à l'article: Sur les AR dédiées, seuls sont transférés l'en-tête de FAL et les données d'utilisateur.

**3.6.42**
**appareil**
équipement matériel physique relié à la liaison

Note 1 à l'article: Un appareil peut contenir plus d'un nœud.

**3.6.43**
**profil d'appareil**
ensemble d'informations et de fonctionnalité dépendant de l'appareil qui assure la cohérence entre des appareils similaires relevant du même type d'appareil

**3.6.44**
**informations de diagnostic**
toutes les données disponibles au niveau du serveur pour des besoins de maintenance

**3.6.45**
**regroupement d'informations de diagnostic**
informations de diagnostic système qui sont regroupées du côté client

**3.6.46**
**AR dynamique**
AR qui requiert l'utilisation de la procédure d'établissement d'AR pour la mettre dans un état établi

**3.6.47**
**nœud d'extrémité**
nœud producteur ou consommateur

**3.6.48**
**point d'extrémité**
l'une des entités en communication impliquées dans une connexion

**3.6.49**
**ingénierie**
terme abstrait qui caractérise l'application ou l'appareil client qui a la responsabilité de configurer un système d'automation par le biais d'éléments de données d'interconnexion

**3.6.50**
**erreur**
discordance entre une valeur ou un état calculé(e), observé(e) ou mesuré(e) et la valeur ou l'état spécifié(e) ou théoriquement correct(e)

**3.6.51**
**classe d'erreurs**
regroupement général pour des définitions d'erreurs connexes et des codes d'erreurs correspondants

**3.6.52**
**code d'erreur**
identification d'un type spécifique d'erreur dans une classe d'erreurs

**3.6.53**
**événement**
instance d'un changement de conditions

**3.6.54**
**sous-réseau FAL**
sous-réseaux constitués d'un ou plusieurs segments de liaison de données, identifiés par un sous-ensemble de l'adresse réseau

Note 1 à l'article: Les sous-réseaux FAL sont autorisés à contenir des ponts, mais pas des routeurs.

**3.6.55**
**variable FIFO**
classe "Variable Object" d'objets variables composée d'un jeu d'éléments de type homogène, dans laquelle le premier élément écrit est le premier élément qui peut être lu

Note 1 à l'article: Dans le bus de terrain, un seul élément commun peut être transféré suite à une invocation de service.

**3.6.56**
**trame**
synonyme déconseillé de DLPDU

**3.6.57**
**freeze**
fonction au niveau des esclaves DP pour le transfert simultané de données entre l'objet de données d'entrée et le processus

**3.6.58**
**groupe**
a) <généralités> terme général pour un ensemble d'objets

Usages spécifiques:

b) <adressage> lors de la description d'une adresse, adresse qui identifie plus d'une entité

**3.6.59**
**interface**
a) frontière partagée entre deux unités fonctionnelles, définies par les caractéristiques fonctionnelles, caractéristiques de signal ou autres caractéristiques selon le cas approprié

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

**3.6.60**
**langage de définition d'interface**
syntaxe et sémantique consistant à décrire de manière formelle les paramètres de services

Note 1 à l'article: Cette description est l'entrée pour le modèle ORPC, notamment pour "l'ORPC wire protocol" (protocole filaire ORPC).

**3.6.61**
**pointeur d'interface**
attribut-clé qui adresse sans ambiguïté une instance d'interface d'objets

**3.6.62**
**invocation**
acte consistant à utiliser un service ou une autre ressource d'un processus d'application

Note 1 à l'article: Chaque invocation représente un processus distinct de commande qui peut être décrit par son contexte. Une fois que le service s'achève ou que l'utilisation de la ressource est libérée, l'invocation cesse d'exister. Pour des invocations de service, un service qui a été lancé, mais n'est pas encore achevé s'appelle invocation de service en cours.

**3.6.63**
**données E/S**
objet désigné pour être transféré de façon cyclique pour des besoins de traitement

**3.6.64**
**diagnostic relatif à l'identificateur**
informations dédiées aux modules pour des besoins de maintenance

**3.6.65**
**indice**
adresse d'un objet au sein d'un processus d'application

**3.6.66**
**instance**
occurrence physique réelle d'un objet au sein d'une classe qui identifie l'un des plusieurs objets au sein de la même classe d'objets

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

Note 1 à l'article: Les termes "objet", "instance" et "instance d'objet" sont utilisés pour se référer à une instance spécifique.

**3.6.67**
**attribut d'instance**
attribut qui est propre à une instance d'objet et n'est pas partagé par la classe d'objets

**3.6.68**
**instancié**
objet qui a été créé dans un appareil

**3.6.69**
**appareil logique**
certaine classe de la FAL qui est une abstraction d'un composant logiciel ou d'un composant de firmware comme une fonction monobloc autonome d'un appareil d'automation

**3.6.70**
**ID de fabricant**
identification de chaque fabrication de produit par un numéro unique

**3.6.71**
**informations de gestion**
informations accessibles par le réseau qui appuient la gestion du fonctionnement du système de bus de terrain, y compris la couche application

Note 1 à l'article: La gestion inclut les fonctions telles que la commande, la surveillance et le diagnostic.

**3.6.72**
**membre**
élément d'un attribut qui est structuré comme étant un élément d'une matrice

**3.6.73**
**routeur de message**
objet au sein d'un nœud qui distribue les demandes de messagerie vers les objets d'application appropriés

**3.6.74**
**méthode**
<objet> synonyme pour un service opérationnel qui est fourni par l'ASE serveur et invoqué par un client

**3.6.75**
**module**
a) <général> composant matériel ou logiciel d'un appareil physique

b) <Type 3> unité adressable à l'intérieur de l'esclave DP

**3.6.76**
**connexion multipoint**
connexion d'un nœud à plusieurs autres nœuds

Note 1 à l'article: Les connexions multipoint permettent que les messages issus d'un seul éditeur soient reçus par plusieurs nœuds consommateurs.

**3.6.77**
**réseau**
ensemble de nœuds reliés par un certain type de support de communication, notamment des répétiteurs intermédiaires, ponts, routeurs et passerelles de couche inférieure

**3.6.78**
**objet**
représentation abstraite d'un composant particulier au sein d'un appareil, habituellement un ensemble de données connexes (sous la forme de variables) et de méthodes (procédures) pour effectuer des opérations sur les données en question qui ont une interface et un comportement clairement définis

**3.6.79**
**appel de procédure distante orientée objet**
modèle pour l'invocation d'une méthode distante orientée objet ou basée sur des composants

**3.6.80**
**service spécifique à un objet**
service propre à la classe d'objets qui le définit

**3.6.81**
**émetteur, origine**
client chargé d'établir un chemin de communication vers une cible

**3.6.82**
**homologue**
rôle d'un point d'extrémité de l'AR qui est capable d'agir tant comme client que comme serveur

**3.6.83**
**appareil physique**
équipement d'automatisation ou autre appareil de réseau

**3.6.84**
**connexion point à point**
connexion qui existe entre exactement deux objets d'application

**3.6.85**
**point d'extrémité d'AR prédéfini**
point d'extrémité de l'AR qui est défini localement dans l'appareil sans utilisation du service "create" (création)

Note 1 à l'article: Les relations AR prédéfinies qui ne sont pas préétablies sont établies avant d'être utilisées.

**3.6.86**
**point d'extrémité d'AR préétabli**
point d'extrémité de l'AR qui est mis dans un état établi pendant la configuration des AE qui commandent ses points d'extrémité

**3.6.87**
**données de processus**
objet(s) déjà prétraité(s) et transféré(s) de façon acyclique pour des besoins d'informations ou de traitement ultérieur

**3.6.88**
**produire**
acte consistant à envoyer des données devant être reçues par un consommateur

**3.6.89**
**producteur**
nœud qui est chargé d'envoyer des données

**3.6.90**
**propriété**
terme général pour les informations descriptives relatives à un objet

**3.6.91**
**fournisseur**
source d'une connexion de données

**3.6.92**
**providerID**
identificateur non ambigu au sein du domaine d'application de l'ACCO attribué par le fournisseur pour reconnaître les données internes d'une source d'interconnexion configurée

**3.6.93**
**éditeur**
rôle d'un point d'extrémité de l'AR qui émet des APDU sur le bus de terrain en vue de leur consommation par un ou plusieurs abonnés

Note 1 à l'article: Un éditeur peut ne pas connaître l'identité ou le nombre d'abonnés et il peut éditer ses APDU en utilisant une AR dédiée

**3.6.94**
**gestionnaire d'édition**
rôle d'un point d'extrémité de l'AR dans lequel il émet une ou plusieurs APDU de demande de services confirmés vers un éditeur pour demander à l'éditeur d'éditer un objet spécifié

Note 1 à l'article: La présente norme définit deux types de gestionnaires d'édition, à savoir les gestionnaires d'édition en mode tirage ("pull") et les gestionnaires d'édition en mode poussée ("push"), chacun de ceux-ci étant défini séparément.

**3.6.95**
**éditeur "pull"**
type d'éditeur qui édite un objet en réponse à une demande reçue de son gestionnaire d'édition "push"

**3.6.96**
**gestionnaire d'édition "pull"**
type de gestionnaire d'édition qui demande qu'un objet spécifié soit édité dans une APDU de réponse correspondante

**3.6.97**
**éditeur "push"**
type d'éditeur qui édite un objet dans une APDU de demande de services non confirmés

**3.6.98**
**gestionnaire d'édition "push"**
type de gestionnaire d'édition qui demande qu'un objet spécifié soit publié en utilisant un service non confirmé

**3.6.99**
**abonné "pull"**
type d'abonné qui reconnaît les APDU de réponse de services non confirmés reçues comme étant des données d'objet produites

**3.6.100**
**abonné "push"**
type d'abonné qui reconnaît les APDU de demande de services non confirmés reçues comme étant des données d'objet produites

**3.6.101**
**code de qualité**
information de statut complémentaire relative à un élément de données

**3.6.102**
**ressource**
capacité de traitement ou d'informations d'un sous-système

**3.6.103**
**point d'extrémité de chemin**
conteneur d'objets contenant des Variable Object (objets variables) d'une classe variable

**3.6.104**
**modèle d'objets exécutables**
ensemble d'objets qui existent dans un appareil conjointement à leurs interfaces et méthodes qui sont accessibles

**3.6.105**
**serveur**
a) rôle d'un AREP dans lequel il retourne une APDU de réponse de service confirmé au client qui a émis la demande

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

**3.6.106**
**service**
opération ou fonction accomplie par un objet et/ou une classe d'objets à la demande d'un autre objet et/ou d'une autre classe d'objets

**3.6.107**
**position**
adresse d'un module au sein d'un esclave DP

**3.6.108**
**abonné**
rôle d'un AREP dans lequel il reçoit des unités APDU produites par un éditeur

**3.6.109**
**sync**
fonction au niveau des esclaves DP pour le transfert simultané de données entre l'objet de données de sortie et le processus

**3.6.110**
**cible**
nœud d'extrémité avec lequel une connexion est établie

**3.6.111**
**service non connecté**
service de messagerie qui ne repose pas sur l'établissement d'une connexion entre appareils avant d'autoriser des échanges d'informations

## 3.7 Abréviations et symboles

| | |
|---|---|
| ACCO | Active Connection Control Object (Objet actif de commande de connexion) |
| AE | Application entity (Entité d'application) |
| AL | Application Layer (Couche application) |
| ALME | Application Layer Management Entity (Entité de gestion de couche application) |
| ALP | Application Layer Protocol (Protocole de couche application) |
| APO | Application Object (Objet d'application) |
| AP | Application Process (Processus d'application) |
| APDU | Application Protocol Data Unit (Unité de données de protocole d'application) |
| API | Application Process Identifier (Interface de Programmation d'Application) |
| AR | Application Relationship (Relation entre applications) |

AREP Application Relationship End Point (Point d'extrémité des relations entre applications)

ASCII American Standard Code for Information Interchange (Code américain normalisé pour l'échange d'information)

ASE Application Service Element (Élément de service d'application)

CID Connection ID (identificateur de connexion)

CIM Computer Integrated Manufacturing (Productique)

CIP Control and Information Protocol (Protocole de commande et d'information)

CM_API Actual Packet Interval (Intervalle réel entre paquets)

CM_RPI Requested Packet Interval (Intervalle requis entre paquets)

Cnf Confirmation

COR Connection originator (Origine de connexion)

CR Communication Relationship (Relation de communication)

CREP Communication Relationship End Point (Point d'extrémité des relations de communication)

DL Préfixe désignant la couche Liaison de données

DLC Data-link Connection (Connexion de liaison de données)

DLCEP Data-link Connection End Point (Point d'extrémité des connexions de liaison de données)

DLL Data Link Layer (Couche liaison de données)

DLM Data Link Management (Gestion de la liaison de données)

DLSAP Data link Service Access Point (Point d'accès au service liaison de données)

DLSDU DL-service-data-unit (Unité de données de service liaison de données)

DNS Domain Name Service (Service de noms de domaine)

DP Decentralised Peripherals (Périphériques décentralisés)

FAL Fieldbus application layer (Couche application de bus de terrain)

FIFO First In First Out (Premier entré, premier sorti)

IHM Interface homme-machine

ID Identificateur

IDL Interface Definition Language (Langage de définition d'interface)

CEI Commission Électrotechnique Internationale

Ind Indication

IP Internet Protocol (Protocole internet)

ISO Organisation internationale de normalisation

LDev Logical Device (Appareil logique)

LME Layer Management Entity (Entité de gestion de couche)

O2T Originator to target (émetteur vers cible) (caractéristique de connexion)

O⇒T Target to originator (Cible vers émetteur) (caractéristique de connexion)

ORPC Object Remote Procedure Call (Appel de procédure distante orientée objet)

| | |
|---|---|
| OSI | Open Systems Interconnect (Interconnexion des systèmes ouverts) |
| PDev | Physical Device (Appareil physique) |
| PDU | Protocol Data Unit (Unité de données de protocole) |
| PL | Physical Layer (Couche physique) |
| QoS | Quality of Service (Qualité de service) |
| QC | Quality Code (Code de qualité) |
| REP | Route Endpoint (Point d'extrémité de chemin) |
| Req | Request (Demande) |
| Rsp | Response (Réponse) |
| RT | Runtime (temps d'exécution) |
| SAP | Service Access Point (Point d'accès au service) |
| SCL | Security Level (Niveau de sécurité) |
| SDU | Service Data Unit (Unité de données de service) |
| SEM | State event matrix (Matrice d'événements d'états) |
| SMIB | System Management Information Base (Base d'informations de gestion de système) |
| SMK | System Management Kernel ((Noyau de gestion système)) |
| STD | State transition diagram (Diagramme des passages d'états), utilisé pour décrire le comportement d'objet |
| S-VFD | Simple Virtual Field Device (Appareil de champ virtuel simple) |
| T2O | Target to originator (Cible vers émetteur) (caractéristique de connexion) |
| T⇒O | Target to originator (Cible vers émetteur) (caractéristique de connexion) |
| VAO | Variable Object (Objet variable) |

## 3.8 Conventions

### 3.8.1 Vue d'ensemble

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

La spécification de classe définit les attributs de la classe. Les attributs sont accessibles à partir d'instances de la classe en utilisant les services d'ASE de gestion d'objets spécifiés à l'Article 5 de la présente norme. La spécification de services définit les services qui sont fournis par l'ASE.

### 3.8.2 Conventions pour les définitions de classes

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

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **Nom de l'ASE** |
| **CLASS:** | | | **Nom de la classe** |
| **CLASS ID:** | | | # |
| **PARENT CLASS:** | | | Nom de la classe parent |
| **ATTRIBUTES:** | | | |
| 1 | (o) | Attribut clé: | identificateur numérique |
| 2 | (o) | Attribut clé: | name |
| 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) La rubrique "FAL ASE:" est le nom de l'élément ASE de la couche FAL (FAL ASE) qui fournit les services pour la classe spécifiée.

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

(3) La rubrique "CLASS ID:" est un numéro qui identifie la classe spécifiée. Ce numéro est unique au sein du FAL ASE qui fournira les services pour cette classe. Lorsqu'il est qualifié par l'identité de son FAL ASE, 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 2048 sont alloués pour identifier les classes définies par l'utilisateur.

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

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

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

a) Chacune des entrées d'attribut contient un numéro de ligne dans la colonne 1, un indicateur obligatoire (m) / facultatif (o) / conditionnel (c) / 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 l'attribut clé.

c) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par période. 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 est à sélectionner 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 utilise les conventions descriptives données 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 qui sont appropriés à la spécification de protocole ou à la spécification d'interface de programmation ou à la spécification de mise en œuvre, mais pas à une définition de service abstrait.

La présente norme utilise un format de tableau pour décrire les paramètres de composants des primitives de service Les paramètres qui s'appliquent à chaque groupe de primitives de service sont consignés en tableaux dans toute la suite 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 les sens de transfert de paramètres utilisés par le service. Les six colonnes possibles sont

1) le nom de paramètre;
2) les paramètres d'entrée de la primitive "request";
3) les paramètres de sortie de la primitive "request";

NOTE 2 Il s'agit d'une fonctionnalité rarement utilisée. Sauf spécification contraire, les paramètres de primitive 'request' sont des paramètres d'entrée.

4) les paramètres de sortie de la primitive " indication";
5) les paramètres d'entrée de la primitive "response"; et

6) les paramètres de sortie de la primitive "confirm".

NOTE 3 Les primitives "request", "indication", "response" et "confirm" sont aussi respectivement appelées 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 rangée de chaque tableau. Dans les colonnes appropriées de la primitive de service, un code est utilisé pour spécifier le type d'usage du paramètre sur la primitive spécifiée dans la colonne:

M Le paramètre est obligatoire pour la primitive;

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

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

– (blanc/vide) le paramètre n'est jamais présent.

S Le paramètre est un élément sélectionné.

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

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

b) une indication qu'une certaine note s'applique à l'entrée;
"(n)" indique que la note "n" suivante contient des informations complémentaires relatives au paramètre et à son utilisation.

#### 3.8.3.3 Procédures de service

Les procédures sont définies en termes des

- interactions entre entités d'application par l'échange d'unités de données de protocole d'application de bus de terrain, et
- interactions entre un fournisseur de services de couche application et un utilisateur de service de couche application dans le même système par l'invocation des 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 série de normes CEI 61158-5 définit des jeux de services abstraits. Il ne s'agit ni de spécifications de protocole ni de spécifications de mise en œuvre ni de spécifications d'interface de programmation concrète. Par conséquent, il y a des restrictions concernant la mesure dans laquelle les procédures de service peuvent être mandatées dans les parties de la CEI 61158-5. Les aspects de protocole qui peuvent varier parmi différentes spécifications de protocole ou différentes mises en œuvre qui instancient les mêmes services abstraits ne sont pas appropriés à être inclus dans ces définitions de service, excepté au niveau d'abstraction qui est nécessairement commun à toutes ces expressions.

Par exemple, le moyen par lequel des fournisseurs de service apparient des PDU de demande et de réponse est approprié pour la spécification dans une norme de spécification de protocole de la CEI 61158-6 mais pas dans une norme de définition de services abstraits de la CEI 61158-5. De même, les méthodes de mise en œuvre locales par lesquelles un fournisseur de service ou utilisateur de service apparie des primitives request et confirm, ou des primitives indication et response, sont appropriées pour une spécification de mise en œuvre ou pour une spécification d'interface de programmation, mais pas pour une norme de service abstrait ou pour une norme de protocole, excepté à un niveau d'abstraction qui est nécessairement commun à tous les modes de réalisation 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.

Des informations supplémentaires sur les procédures de service conceptuelles d'une mise en œuvre d'un protocole qui réalise les services de l'une des définitions de service abstrait de la CEI 61158-5 peuvent être consultées 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 détaillés dans la CEI 61158-1, Article 9.

# 5 ASE Data type

## 5.1 Vue d'ensemble

Une vue d'ensemble de l'ASE Data type (type de données) et des relations entre types de données est fournie dans la CEI 61158-1, 10.1.

## 5.2 Définition formelle des objets data type (types de données)

### 5.2.1 Classe de types de données

#### 5.2.1.1 Modèle

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

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE DATA TYPE** |
| **CLASS:** | | | **DATA TYPE** |
| **CLASS ID:** | | | 5 (FIXED LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY) |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut-clé: | Data type Numeric Identifier |
| 2 | (o) | Attribut clé: | 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 (type de données de champ) |
| 6 | (c) | Contrainte: | Format = ARRAY |
| 6.1 | (m) | Attribut: | Number of Array Elements |
| 6.2 | (m) | Attribut: | Array Element Data type (type de données d'élément de matrice) |

#### 5.2.1.2 Attributs

**Data type Numeric Identifier**

Cet attribut identifie l'identificateur numérique du type de données connexe.

**Data type Name**

Cet attribut facultatif identifie le nom du type de données connexe.

**Format**

Cet attribut identifie le type de données comme étant fixed-length, string, array ou structure de données (data structure).

**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" ou "STRING". Pour les types de données FIXED LENGTH, il représente la longueur en octets. Pour les types de données STRING, il représente la longueur en octets pour un élément simple d'une chaîne.

**Number of Fields**

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 donnée 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 (c'est-à-dire: identificateur numérique), soit en intégrant ici une définition de type de données construit. Pour intégrer une description, il est utilisé la description de "Embedded Data type" (type de données intégré) qui est montrée ci-après.

**Number of Array Elements**

Cet attribut conditionnel définit le nombre d'éléments de type array (matrice, tableau). Les éléments de matrice sont indicés 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".

**Array Element Data type**

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

**Embedded Data type Description**

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 montrés dans le modèle sont définis ci-dessus dans la classe de types de données, à l'exception de l'attribut "Embedded Data type" (type de données intégré), qui est une référence récursive à cet attribut. Il est utilisé pour définir des éléments imbriqués.

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 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 Types Boolean (booléens)

##### 5.3.1.1.1 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 |

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

##### 5.3.1.1.2 BOOL

Ce type de la CEI 61131-3 est le même que le type Boolean.

##### 5.3.1.1.3 VT_BOOLEAN

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | VT_BOOLEAN |
| 4 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type de données exprime un type de données Boolean avec les valeurs TRUE (-1) et FALSE (0) (voir Interger16).

#### 5.3.1.2 Types Bitstring (chaîne de bits)

##### 5.3.1.2.1 BitString8

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 22 |
| 2 | Data type Name | = | Bitstring8 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 1 |

Ce type contient un élément de type BitString.

##### 5.3.1.2.2 OCTET

Ce type de la CEI 61131-3 est le même que le type Bitstring8.

##### 5.3.1.2.3 BitString16

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 23 |
| 2 | Data type Name | = | Bitstring16 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 2 |

##### 5.3.1.2.4 WORD

Ce type de la CEI 61131-3 est le même que le type Bitstring16.

#### 5.3.1.2.5 BitString32

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 24 |
| 2 | Data type Name | = | Bitstring32 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 4 |

#### 5.3.1.2.6 DWORD

Ce type de la CEI 61131-3 est le même que le type Bitstring32.

#### 5.3.1.2.7 BitString64

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 57 |
| 2 | Data type Name | = | Bitstring64 |
| 3 | Format | = | FIXED LENGTH |
| 5.1 | Octet Length | = | 8 |

#### 5.3.1.2.8 LWORD

Ce type de la CEI 61131-3 est le même que le type Bitstring64.

### 5.3.1.3 Types Currency

#### 5.3.1.3.1 currency

| | | | |
|---|---|---|---|
| **CLASS:** | **Data type** | | |
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | currency |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

Ce data type définit un entier signé de 64 bits en unités de 1/10 000 (ou 1/100 de centime). Nombre de type currency stocké sous forme d'un nombre entier de 8 octets en complément à deux, normalisé par 10 000 pour donner un nombre en virgule fixe avec 15 chiffres à la gauche de la virgule décimale et 4 chiffres à droite. Cette représentation fournit une étendue de ±922337203685477,5807. Ce data type est utile pour les calculs impliquant l'argent ou pour tout calcul en virgule fixe où l'exactitude est particulièrement importante.

### 5.3.1.4 Types Date

#### 5.3.1.4.1 BinaryDate

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 11 |
| 2 | Data type Name | = | BinaryDate |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 7 |

Ce type de données est constitué de six éléments de valeurs unsigned (non signées) et exprime l'heure et la date calendaires. Le premier élément est un type de données Unsigned16 et donne la fraction d'une minute en millisecondes. Le deuxième élément est un type de données Unsigned8 et donne la fraction d'une heure en minutes. Le troisième élément est un type de données Unsigned8 et donne la fraction d'un jour en heures. Le quatrième élément est un type de données Unsigned8. Ses trois (3) bits de poids fort donnent le jour de la semaine et ses cinq (5) bits de poids faible donnent le jour du mois. Le

cinquième élément est un type de données Unsigned8 et donne le mois. Le dernier élément est un type de données Unsigned8 et donne l'année.

#### 5.3.1.4.2 BinaryDate2000

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 51 |
| 2 | Data type Name | = | BinaryDate2000 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

Ce type de données est constitué de six éléments de valeurs unsigned (non signées) et exprime l'heure et la date calendaires. Le premier élément est un type de données Unsigned16 et donne la fraction d'une minute en millisecondes. Le deuxième élément est un type de données Unsigned8 et donne la fraction d'une heure en minutes. Le troisième élément est un type de données Unsigned8 et donne la fraction d'un jour en heures. Le quatrième élément est un type de données Unsigned8. Ses trois (3) bits de poids fort donnent le jour de la semaine et ses cinq (5) bits de poids faible donnent le jour du mois. Le cinquième élément est un type de données Unsigned8 et donne le mois. Le dernier élément est un type de données Unsigned16 et donne l'année.

#### 5.3.1.4.3 Date

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 50 |
| 2 | Data type Name | = | Date |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 7 |

Ce type de données est constitué de six éléments de valeurs unsigned (non signées) et exprime l'heure et la date calendaires. Le premier élément est un type de données Unsigned16 et donne la fraction d'une minute en millisecondes. Le deuxième élément est un type de données Unsigned8 et donne la fraction d'une heure en minutes. Le troisième élément est un type de données Unsigned8 et donne la fraction d'un jour en heures, le bit de poids fort indiquant l'heure normalisée (Standard Time) ou l'heure d'économie d'énergie (Daylight Saving Time). Le quatrième élément est un type de données Unsigned8. Ses trois (3) bits de poids fort donnent le jour de la semaine et ses cinq (5) bits de poids faible donnent le jour du mois. Le cinquième élément est un type de données Unsigned8 et donne le mois. Le dernier élément est un type de données Unsigned8 et donne l'année. Les valeurs 0 ... 50 correspondent aux années 2000 à 2050, les valeurs 51 ... 99 correspondent aux années 1951 à 1999.

#### 5.3.1.4.4 DATE

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | DATE |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type de la CEI 61131-3 est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type unsigned (non signé) a une longueur de deux octets. Il exprime la date comme un nombre de jours, commençant à partir de 1972.01.01 (1er janvier 1972), début de l'ère du temps universel coordonné (TUC), jusqu'au 2151.06.06 (6 juin 2151), c'est-à-dire une plage totale de 65 536 jours.

#### 5.3.1.4.5 date

Ce type de données est le même que le type Float64.

Le type de données date a une résolution de l'ordre d'une nanoseconde. Il est valide pour les dates entre le 1er janvier 0100 et le 31 décembre 9999. La valeur 0,0 a été définie pour le 30 décembre 1899, 00:00. La partie entière de la valeur représente les jours après le 30 décembre 1899 (pour les dates antérieures à ce jour, la valeur correspondante est négative); la partie fractionnaire définit l'heure du jour en question.

#### 5.3.1.4.6 TimeOfDay

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

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

#### 5.3.1.4.7 TimeOfDay with date indication (avec indication de date)

Ce type de données est le même que le type de données TimeOfDay défini ci-dessus.

#### 5.3.1.4.8 TimeOfDay without date indication (sans indication de date)

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 52 |
| 2 | Data type Name | = | TimeOfDay without date indication (sans indication de date) |
| 4 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type de données est constitué d'un seul élément d'une valeur non signée et exprime l'heure du jour. L'élément est un type de données Unsigned32 et donne l'heure après minuit en millisecondes.

#### 5.3.1.4.9 TIME_OF_DAY

Ce type de la CEI 61131-3 est le même que le type TimeofDay without date indication.

#### 5.3.1.4.10 TimeDifference

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

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

#### 5.3.1.4.11 TimeDifference with date indication (avec indication de date)

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 53 |
| 2 | Data type Name | = | TimeDifference with date indication |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 6 |

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

#### 5.3.1.4.12 TimeDifference without date indication (sans indication de date)

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 54 |
| 2 | Data type Name | = | TimeDifference without date indication |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type de données est constitué d'un seul élément d'une valeur unsigned (non signée) qui exprime la différence de temps. L'élément est un type de données Unsigned32 qui donne la partie fractionnaire d'un jour en millisecondes.

#### 5.3.1.4.13 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 |

Ce type simple exprime l'heure ou la différence temporelle en un nombre binaire en complément à deux avec une longueur de huit octets. L'unité de temps est 1/32 milliseconde.

#### 5.3.1.4.14 UniversalTime

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 16 |
| 2 | Data type Name | = | UniversalTime |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 12 |

Ce type simple est constitué de douze éléments de type VisibleString. (YYMMDDHHMMSS). Il est le même que celui défini dans l'ISO/CEI 8824, excepté que le différentiel de temps local n'est pas pris en charge.

#### 5.3.1.4.15 FieldbusTime

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 17 |
| 2 | Data type Name | = | FieldbusTime |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 7 |

Ce data type est défini dans la présente norme comme DL-Time (temps DL).

#### 5.3.1.5 Types Enumerated (énumérés)

##### 5.3.1.5.1 PERSISTDEF

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | PERSISTDEF |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Les valeurs admissibles sont montrées dans le Tableau 1.

**Tableau 1 – PERSISTDEF**

| Value |
|---|
| CBAVolatile |
| CBAPendingPersistent |
| CBAPersistent |

##### 5.3.1.5.2 VARTYPE

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | VARTYPE |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Le type de données VARTYPE spécifie le type de données qui régit l'interprétation des données. Les valeurs admissibles sont montrées dans le Tableau 2.

**Tableau 2 – VARTYPE**

| Value | Data type |
|---|---|
| VT_EMPTY | aucune valeur |
| VT_NULL | valeur vide (aucune donnée valide) |
| VT_BOOL | VT_BOOLEAN |
| VT_I1 | char |
| VT_I2 | short |
| VT_I4 | long |
| VT_UI1 | unsigned char |
| VT_UI2 | unsigned short |
| VT_UI4 | unsigned long |
| VT_R4 | float |
| VT_R8 | double |
| VT_CY | Currency |
| VT_DATE | date |
| VT_BSTR | adresse d'un BSTR |
| VT_SAFEARRAY_BOOL | adresse d'un SAFEARRAY (VT_BOOLEAN) |
| VT_SAFEARRAY_I1 | adresse d'un SAFEARRAY (char) |
| VT_SAFEARRAY_I2 | adresse d'un SAFEARRAY (short) |
| VT_SAFEARRAY_I4 | adresse d'un SAFEARRAY (long) |
| VT_SAFEARRAY_UI1 | adresse d'un SAFEARRAY (unsigned char) spécifiée. |
| VT_SAFEARRAY_UI2 | adresse d'un SAFEARRAY (unsigned short) spécifiée. |

| Value | Data type |
|---|---|
| VT_SAFEARRAY_UI4 | adresse d'un SAFEARRAY (unsigned long) spécifiée. |
| VT_SAFEARRAY_R4 | adresse d'un SAFEARRAY (float) |
| VT_SAFEARRAY_R8 | adresse d'un SAFEARRAY (double) |
| VT_SAFEARRAY_CY | adresse d'un SAFEARRAY (Currency) |
| VT_SAFEARRAY_DATE | adresse d'un SAFEARRAY (date) |
| VT_SAFEARRAY_BSTR | adresse d'un SAFEARRAY (BSTR) |
| VT_DISPATCH | Interface Pointer (Pointeur d'interface) vers une interface IDispatch |
| VT_UNKNOWN | Interface Pointer (Pointeur d'interface) vers une interface IUnknown |
| VT_USERDEFINED | adresse d'une userdefined struct (structure définie par l'utilisateur) |
| VT_ERROR | Un HRESULT est spécifié. |
| Il convient que tous les data types structurés aient le code de type VT_USERDEFINED. | |

#### 5.3.1.5.3 ITEMQUALITYDEF

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name | = ITEMQUALITYDEF |
| 3 | Format | = FIXED LENGTH |
| 4.1 | Octet Length | = 1 |

Ce type de données contient les informations de statut des données connexes. Il consiste en trois parties: Quality (Qualité), Substatus (Sous-statut) et Limits (Limites). Il y a quatre états de quality (Bad – la valeur n'est pas utilisable; Uncertain – la qualité de la valeur est inférieure à la normale, mais la valeur peut encore être utilisable; Good (Non Cascade) – la qualité de la valeur est bonne, des conditions d'alarme éventuelles peuvent être indiquées par le substatus (sous-statut); Good (Cascade) – la valeur peut être utilisée dans la commande), un ensemble de valeurs de sous-statuts, et quatre états des limites (OK – la valeur est libre de varier; low limited (LL, en limite basse) – la valeur a atteint ses limites basses, high limited (HL, en limite haute) – la valeur a atteint ses limites hautes; constant (C) – la valeur ne peut pas varier, quoi que fasse le processus). Les valeurs admissibles sont montrées dans le Tableau 3.

**Tableau 3 – ITEMQUALITYDEF**

| Qualité | Valeur | Description |
|---|---|---|
| Bad | BadNonSpecific | Il n'y a pas de raison spécifique expliquant pourquoi la valeur est bad (mauvaise). Utilisée pour la propagation. |
| | BadNonSpecificLL | et en limite basse |
| | BadNonSpecificHL | et en limite haute |
| | BadNonSpecificC | et constante |
| | BadConfigurationError | Positionné si la valeur n'est pas utilisable car le bloc présente un problème, selon ce qu'un éditeur spécifique peut détecter. |
| | BadConfigurationErrorLL | et en limite basse |
| | BadConfigurationErrorHL | et en limite haute |
| | BadConfigurationErrorC | et constante |
| | BadNotConnected | Positionné si cette entrée est à connecter et n'est pas connectée. |
| | BadNotConnectedLL | et en limite basse |
| | BadNotConnectedHL | et en limite haute |
| | BadNotConnectedC | et constante |

| Qualité | Valeur | Description |
|---|---|---|
| | BadDeviceFailure | Positionné si la source de la valeur est altérée par une défaillance d'appareil. |
| | BadDeviceFailureLL | et en limite basse |
| | BadDeviceFailureHL | et en limite haute |
| | BadDeviceFailureC | et constante |
| | BadSensorFailure | Positionné si l'appareil peut déterminer cette condition. Les limites définissent le sens qui a été dépassé. |
| | BadSensorFailureLL | et en limite basse |
| | BadSensorFailureHL | et en limite haute |
| | BadSensorFailureC | et constante |
| | BadLastKnownValue | Positionné si la valeur a été établie par une communication, qui a maintenant échoué. |
| | BadLastKnownValueLL | et en limite basse |
| | BadLastKnownValueHL | et en limite haute |
| | BadLastKnownValueC | et constante |
| | BadCommFailure | Positionné s'il n'y a jamais eu de communication avec la valeur de la dernière fois lorsqu'elle a été Out of Service (hors service). |
| | BadCommFailureLL | et en limite basse |
| | BadCommFailureHL | et en limite haute |
| | BadCommFailureC | et constante |
| | BadOutOfService | La valeur n'est pas fiable car le bloc n'est pas en cours d'évaluation, et peut être en construction par un outil de configuration. Elle est positionnée si le mode bloc est O/S (hors service). |
| | BadOutOfServiceLL | et en limite basse |
| | BadOutOfServiceHL | et en limite haute |
| | BadOutOfServiceC | et constante |
| Uncertain | UncertainNonSpecific | Il n'y a pas de raison spécifique expliquant que la valeur soit incertaine. Utilisée pour la propagation. |
| | UncertainNonSpecificLL | et en limite basse |
| | UncertainNonSpecificHL | et en limite haute |
| | UncertainNonSpecificC | et constante |
| | UncertainLastUsableValue | Ce qui écrivait cette valeur, quel qu'il soit, a cessé de le faire. Cela est utilisé pour le traitement de sécurité intégrée. |
| | UncertainLastUsableValueLL | et en limite basse |
| | UncertainLastUsableValueHL | et en limite haute |
| | UncertainLastUsableValueC | et constante |
| | UncertainSubstituteSet | Une valeur prédéfinie est utilisée en lieu et place de la valeur calculée. Cela est utilisé pour le traitement de sécurité intégrée. |
| | UncertainSubstituteSetLL | et en limite basse |
| | UncertainSubstituteSetHL | et en limite haute |
| | UncertainSubstituteSetC | et constante |
| | UncertainInitialValue | Valeur de paramètres volatils pendant et après la réinitialisation de l'appareil ou d'un paramètre. |
| | UncertainInitialValueLL | et en limite basse |
| | UncertainInitialValueHL | et en limite haute |

| Qualité | Valeur | Description |
|---|---|---|
| | UncertainInitialValueC | et constante |
| | UncertainSensorNotAccurate | Positionné si la valeur est à l'une des limites du capteur. Les limites définissent le sens qui a été dépassé. Également positionné si l'appareil peut déterminer que le capteur a une exactitude réduite (par exemple: analyseur dégradé), auquel cas aucune limite n'est fixée. |
| | UncertainSensorNotAccurateLL | et en limite basse |
| | UncertainSensorNotAccurateHL | et en limite haute |
| | UncertainSensorNotAccurateC | et constante |
| | UncertainEngineeringUnitsExceeded | Positionné si la valeur se situe hors de la plage des valeurs définies pour ce paramètre. Les limites définissent le sens qui a été dépassé. |
| | UncertainEngineeringUnitsExceededLL | et en limite basse |
| | UncertainEngineeringUnitsExceededHL | et en limite haute |
| | UncertainEngineeringUnitsExceededC | et constante |
| | UncertainSubNormal | Positionné si une valeur dérivée de plusieurs valeurs a moins que le nombre requis de sources Good (c'est-à-dire: bonnes). |
| | UncertainSubNormalLL | et en limite basse |
| | UncertainSubNormalHL | et en limite haute |
| | UncertainSubNormalC | et constante |
| | UncertainConfigurationError | Positionné s'il y a incohérence concernant la paramétrisation ou la configuration, selon ce qu'un éditeur spécifique peut détecter. |
| | UncertainConfigurationErrorLL | et en limite basse |
| | UncertainConfigurationErrorHL | et en limite haute |
| | UncertainConfigurationErrorC | et constante |
| | UncertainSimulatedValue | Positionné lorsque la valeur de processus est écrite par l'opérateur alors que le bloc est en mode manuel. |
| | UncertainSimulatedValueLL | et en limite basse |
| | UncertainSimulatedValueHL | et en limite haute |
| | UncertainSimulatedValueC | et constante |
| | UncertainSensorCalibration | Positionné au cours du processus d'étalonnage actif ensemble avec la valeur mesurée courante. |
| | UncertainSensorCalibrationLL | et en limite basse |
| | UncertainSensorCalibrationHL | et en limite haute |
| | UncertainSensorCalibrationC | et constante |
| Good | GoodNonCascOk | Aucune erreur ou condition spéciale n'est associée à cette valeur. |
| Non Cascade | GoodNonCascOkC | et constante |
| | GoodNonCascActiveUpdateEvent | Positionné si la valeur est good (bonne) et le bloc a un "Update Event" actif (événement de mise à jour actif). |
| | GoodNonCascActiveUpdateEventLL | et en limite basse |
| | GoodNonCascActiveUpdateEventHL | et en limite haute |
| | GoodNonCascActiveUpdateEventC | et constante |
| | GoodNonCascActiveAdvisoryAlarm | Positionné si la valeur est bonne et le bloc a une Alarm active avec une priorité inférieure à 8. |
| | GoodNonCascActiveAdvisoryAlarmLL | et en limite basse |
| | GoodNonCascActiveAdvisoryAlarmHL | et en limite haute |
| | GoodNonCascActiveAdvisoryAlarmC | et constante |

| Qualité | Valeur | Description |
|---|---|---|
| | GoodNonCascActiveCriticalAlarm | Positionné si la valeur est bonne et le bloc a une Alarm active avec une priorité supérieure ou égale à 8. |
| | GoodNonCascActiveCriticalAlarmLL | et en limite basse |
| | GoodNonCascActiveCriticalAlarmHL | et en limite haute |
| | GoodNonCascActiveCriticalAlarmC | et constante |
| | GoodNonCascUnackUpdateEvent | Positionné si la valeur est good (bonne) et le bloc a un "Update Event" non acquitté (événement de mise à jour non acquitté). |
| | GoodNonCascUnackUpdateEventLL | et en limite basse |
| | GoodNonCascUnackUpdateEventHL | et en limite haute |
| | GoodNonCascUnackUpdateEventC | et constante |
| | GoodNonCascUnackAdvisoryAlarm | Positionné si la valeur est bonne et le bloc a une Alarm non acquittée avec une priorité inférieure à 8. |
| | GoodNonCascUnackAdvisoryAlarmLL | et en limite basse |
| | GoodNonCascUnackAdvisoryAlarmHL | et en limite haute |
| | GoodNonCascUnackAdvisoryAlarmC | et constante |
| | GoodNonCascUnackCriticalAlarm | Positionné si la valeur est bonne et le bloc a une Alarm non acquittée avec une priorité supérieure ou égale à 8. |
| | GoodNonCascUnackCriticalAlarmLL | et en limite basse |
| | GoodNonCascUnackCriticalAlarmHL | et en limite haute |
| | GoodNonCascUnackCriticalAlarmC | et constante |
| | GoodNonCascInitialFailSafe | Cette valeur provient du bloc qui souhaite que son bloc de sortie suivant (AO par exemple) aille à "Fail Safe". |
| | GoodNonCascInitialFailSafeLL | et en limite basse |
| | GoodNonCascInitialFailSafeHL | et en limite haute |
| | GoodNonCascInitialFailSafeC | et constante |
| | GoodNonCascMaintenanceRequired | L'appareil fonctionne encore sans défaillance, mais bientôt une maintenance sera nécessaire. Cela peut être détecté, par exemple, par un Transducer Block ayant une valeur de pH-mètre. |
| | GoodNonCascMaintenanceRequiredLL | et en limite basse |
| | GoodNonCascMaintenanceRequiredHL | et en limite haute |
| | GoodNonCascMaintenanceRequiredC | et constante |
| Good | GoodCascOk | Aucune erreur ou condition spéciale n'est associée à cette valeur. |
| Cascade | GoodCascOkC | et constante |
| | GoodCascInitializationAcknowledge | La valeur est une valeur initialisée issue d'une source (paramètres cascade input, remote-cascade in, et remote-output in). |
| | GoodCascInitializationAcknowledgeLL | et en limite basse |
| | GoodCascInitializationAcknowledgeHL | et en limite haute |
| | GoodCascInitializationAcknowledgeC | et constante |
| | GoodCascInitializationRequest | La valeur est une valeur initialisée pour une source (paramètre d'entrée de rétrocalcul) car la boucle inférieure est cassée ou le mode est erroné. |
| | GoodCascInitializationRequestLL | et en limite basse |
| | GoodCascInitializationRequestHL | et en limite haute |
| | GoodCascInitializationRequestC | et constante |

| Qualité | Valeur | Description |
|---|---|---|
| | GoodCascNotInvited | La valeur provient d'un bloc qui n'a pas de mode-cible susceptible d'utiliser cette entrée. Cela couvre tous les cas autres que Fail Safe Active, Local Override, et Not Selected. Le mode-cible peut être le prochain mode autorisé de plus de haute priorité dans le cas du délestage d'un calculateur de surveillance. |
| | GoodCascNotInvitedLL | et en limite basse |
| | GoodCascNotInvitedHL | et en limite haute |
| | GoodCascNotInvitedC | et constante |
| | GoodCascDoNotSelect | La valeur provient d'un bloc qu'il ne convient pas de sélectionner, en raison de conditions internes ou au-dessus du bloc. |
| | GoodCascDoNotSelectLL | et en limite basse |
| | GoodCascDoNotSelectHL | et en limite haute |
| | GoodCascDoNotSelectC | et constante |
| | GoodCascLocalOverride | La valeur provient d'un bloc qui a été verrouillé par un commutateur à clé local ou qui est un Complex AO/DO avec logique d'interverrouillage active. Il convient de propager la défaillance de commande normale vers un bloc PID pour des besoins d'alarme et d'affichage. Cela sous-entend également "Not Invited". |
| | GoodCascLocalOverrideLL | et en limite basse |
| | GoodCascLocalOverrideHL | et en limite haute |
| | GoodCascLocalOverrideC | et constante |
| | GoodCascInitiateFailSafe | La valeur provient du bloc qui souhaite que son bloc de sortie aval (AO par exemple) aille à "Fail Safe". Cela est déterminé par une option de bloc pour déclencher Fail Safe si le statut de l'entrée primaire et/ou de l'entrée en cascade passe à Bad. |
| | GoodCascInitiateFailSafeLL | et en limite basse |
| | GoodCascInitiateFailSafeHL | et en limite haute |
| | GoodCascInitiateFailSafeC | et constante |

#### 5.3.1.5.4 STATEDEF

**CLASS:** **Data type**
**ATTRIBUTES:**
2 Data type Name = STATEDEF
3 Format = FIXED LENGTH
4.1 Octet Length = 2

Les valeurs admissibles sont montrées dans le Tableau 4.

**Tableau 4 – STATEDEF**

| Value |
|---|
| CBANonExistent |
| CBAInitializing |
| CBAReady |
| CBAOperating |
| CBADefect |

#### 5.3.1.5.5 GROUPERRORDEF

**CLASS: Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | GROUPERRORDEF |
| 3 | Format = | FIXED LENGTH |
| 4.1 | Octet Length = | 2 |

Les valeurs admissibles sont montrées dans le Tableau 5.

**Tableau 5 – GROUPERRORDEF**

| Value |
|---|
| CBANonAccessible |
| CBAOkay |
| CBAProblem |
| CBAUnknown |

#### 5.3.1.5.6 ACCESSRIGHTSDEF

**CLASS: Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | ACCESSRIGHTSDEF |
| 3 | Format = | FIXED LENGTH |
| 4.1 | Octet Length = | 2 |

Les valeurs admissibles sont montrées dans le Tableau 6.

**Tableau 6 – ACCESSRIGHTSDEF**

| Value |
|---|
| CBANoAccess |
| CBAReadAccess |
| CBAWriteAccess |
| CBAFullAccess |

### 5.3.1.6 Types Handle (identification)

#### 5.3.1.6.1 HRESULT

**CLASS: Data type**
**ATTRIBUTES:**

| | | |
|---|---|---|
| 2 | Data type Name = | HRESULT |
| 3 | Format = | FIXED LENGTH |
| 4.1 | Octet Length = | 4 |

Les valeurs admissibles sont montrées dans le Tableau 7. Les valeurs définies de HRESULT peuvent être étendues par des valeurs spécifiques à un utilisateur. Au besoin, il convient d'appliquer le schéma de codage tel que défini dans la CEI 61158-6-5. En particulier, il convient d'utiliser le code de sévérité pour indiquer un succès ou une erreur.

**Tableau 7 – HRESULT**

| Value | Description |
|---|---|
| CBA_E_MALFORMED | L'identificateur est mal formé (trop long, caractères interdits, absence de caractères de séparation, syntaxe non valide). |
| CBA_E_UNKNOWNOBJECT | L'objet spécifié dans l'identificateur d'élément n'est pas connu. |
| CBA_E_UNKNOWNMEMBER | Le membre spécifié dans l'identificateur d'élément n'est pas connu. |
| CBA_E_TYPEMISMATCH | Le type spécifié n'est pas conforme avec le type attendu. |
| CBA_E_INVALIDENUMVALUE | La valeur énumération est non valide. |
| CBA_E_INVALIDID | Le ConsumerID ou le ProviderID n'est pas valide. |
| CBA_E_INVALIDEPSILON | Le type ou la valeur Epsilon n'est pas valide. |
| CBA_E_INVALIDSUBSTITUTE | Le type ou la valeur Substitute (c'est-à-dire: substitut) n'est pas valide. |
| CBA_E_INVALIDCONNECTION | Il n'est pas permis de spécifier une connexion à partir d'un identificateur à soi-même. |
| CBA_E_INVALIDCOOKIE | La valeur Cookie n'est pas valide. |
| CBA_E_TIMEVALUEUNSUPPORTED | La valeur de temps n'est pas prise en charge. |
| CBA_E_QOSTYPEUNSUPPORTED | Le type QoS n'est pas pris en charge. |
| CBA_E_QOSVALUEUNSUPPORTED | La valeur QoS n'est pas prise en charge. |
| CBA_E_PERSISTRUNNING | Pendant que le service Save fonctionne, il n'est permis d'apporter aucun changement à la base de données de configuration (essayer de nouveau dans quelques secondes). |
| CBA_E_INUSE | La destination spécifiée dans l'identificateur d'élément est déjà connectée à un autre fournisseur. |
| CBA_E_NOTAPPLICABLE | L'opération n'est pas applicable actuellement. |
| CBA_E_NONACCESSIBLE | L'élément n'est pas accessible. |
| CBA_E_DEFECT | Défaut matériel détecté, remplacement nécessaire. |
| CBA_S_PERSISTPENDING | La valeur demandée n'est pas persistante actuellement. |
| CBA_S_ESTABLISHING | La connexion n'est pas encore établie. |
| CBA_S_NOCONNECTION | Il n'y a aucune connexion du fournisseur à ce abonné spécifique. |
| CBA_S_VALUEBUFFERED | La valeur a seulement été placée en buffer et n'a pas d'effet immédiat sur le processus (par exemple: l'appareil est dans l'état CBAReady). |
| CBA_S_VALUEUNCERTAIN | La valeur demandée est incertaine actuellement. |
| E_OUTOFMEMORY | Mémoire insuffisante pour parachever l'appel. |
| E_INVALIDARG | Un ou plusieurs arguments ne sont pas valides. |
| E_NOTIMPL | Le service n'est pas mis en œuvre. |
| E_FAIL | Erreur non spécifiée. |
| E_NOINTERFACE | Une telle interface n'est pas prise en charge. |
| RPC_S_PROCNUM_OUT_OF_RANGE | Le numéro de procédure est hors plage. |
| RPC_E_INVALID_OXID | L'exportateur d'objet n'a pas été trouvé. |
| RPC_E_INVALID_OID | L'objet spécifié n'a pas été trouvé ou reconnu. |
| RPC_E_INVALID_SET | L'ensemble d'exportateur d'objet n'a pas été trouvé. |
| RPC_E_INVALID_OBJECT | L'objet demandé n'existe pas. |
| RPC_E_VERSION_MISMATCH | La version ORPC sur la machine client et celle sur la machine serveur ne concordent pas. |
| DISP_E_BADINDEX | Indice non valide. |
| DISP_E_BADPARAMCOUNT | Le nombre d'éléments fournis à DISPPARAMS est différent du nombre d'arguments acceptés par la méthode ou la propriété. |
| DISP_E_BADVARTYPE | L'un des arguments n'est pas un type "variant" valide |

| Value | Description |
|---|---|
| DISP_E_EXCEPTION | L'application a besoin de déclencher une exception. Dans ce cas, il convient de remplir la structure passée dans pExcepInfo. |
| DISP_E_MEMBERNOTFOUND | Le membre demandé n'existe pas ou l'appel à Invoke a tenté d'établir la valeur d'une propriété en lecture seule. |
| DISP_E_NONAMEDARGS | Cette mise en œuvre d'IDispatch ne prend pas en charge les arguments nommés. |
| DISP_E_OVERFLOW | L'un des arguments ne pouvait pas être forcé à être du type spécifié. |
| DISP_E_PARAMNOTFOUND | L'un des paramètres DISPID ne correspond pas à un paramètre sur la méthode. Dans ce cas, il convient de mettre puArgErr au premier argument qui contient l'erreur. |
| DISP_E_TYPEMISMATCH | Un ou plusieurs des arguments ne pouvaient pas être forcés. L'indice au sein de rgvarg du premier paramètre ayant le type incorrect est retourné dans le paramètre puArgErr. |
| DISP_E_UNKNOWNINTERFACE | L'identificateur d'interface passé dans riid n'est pas IID_NULL. |
| DISP_E_UNKNOWNLCID | Langue inconnue. |
| DISP_E_UNKNOWNNAME | Nom inconnu. |
| DISP_E_PARAMNOTOPTIONAL | Un paramètre requis a été omis. |
| TYPE_E_ELEMENTNOTFOUND | Élément introuvable. |
| S_OK | Succès, "everything worked" (tout a fonctionné). |
| S_FALSE | Succès mais avec résultats complémentaires, "function worked and the result is false" ("la fonction a fonctionné et le résultat est faux"). |

#### 5.3.1.7 Types numériques

##### 5.3.1.7.1 BCD

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 35 |
| 2 | Data type Name | = | Unsigned8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 1 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur égale à un octet. Dans ce type, les quatre bits de poids faible sont utilisés pour exprimer une valeur BCD (ou DCB «décimal codé binaire») qui est comprise entre zéro et neuf inclus. Les quatre bits de poids fort ne sont pas utilisés.

##### 5.3.1.7.2 Types Floating Point (virgule flottante)

###### 5.3.1.7.2.1 Float32

**CLASS:** **Data type**

**ATTRIBUTES:**

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

Ce type a une longueur de quatre octets. Le format pour float32 est celui défini par l'ANSI/IEEE 754 comme single precision ("simple précision").

#### 5.3.1.7.2.2 Floating-point (Virgule flottante)

Ce type de données est le même que le type Float32.

#### 5.3.1.7.2.3 REAL

Ce type de la CEI 61131-3 est le même que le type Float32.

#### 5.3.1.7.2.4 float

Ce type de données est le même que le type Float32.

#### 5.3.1.7.2.5 Float64

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 15 |
| 2 | Data type Name | = | Float64 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

Ce type a une longueur de huit octets. Le format pour Float64 est celui défini par l'ANSI/IEEE 754 comme double precision ("double précision").

#### 5.3.1.7.2.6 LREAL

Ce type de la CEI 61131-3 est le même que le type Float64.

#### 5.3.1.7.2.7 double

Ce type de données est le même que le type Float64.

### 5.3.1.7.3 Types entiers

#### 5.3.1.7.3.1 Integer8

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

Ce type integer est un nombre binaire en complément à deux avec une longueur égale à un octet.

#### 5.3.1.7.3.2 SINT

Ce type de la CEI 61131-3 est le même que le type Integer8.

#### 5.3.1.7.3.3 char

Ce type de données est le même que le type Integer8.

#### 5.3.1.7.3.4 Integer16

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 3 |
| 2 | Data type Name | = | Integer16 |

| | | | |
|---|---|---|---|
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type integer est un nombre binaire en complément à deux avec une longueur égale à deux octets.

#### 5.3.1.7.3.5 INT

Ce type de la CEI 61131-3 est le même que le type Integer16.

#### 5.3.1.7.3.6 short

Ce type de données est le même que le type Integer16.

#### 5.3.1.7.3.7 Integer32

**CLASS:** **Data type**

**ATTRIBUTES:**

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

Ce type integer est un nombre binaire en complément à deux avec une longueur égale à quatre octets.

#### 5.3.1.7.3.8 DINT

Ce type de la CEI 61131-3 est le même que le type Integer32.

#### 5.3.1.7.3.9 long

Ce type de données est le même que le type Integer32.

#### 5.3.1.7.3.10 Integer64

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 55 |
| 2 | Data type Name | = | Integer64 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

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

#### 5.3.1.7.3.11 LINT

Ce type de la CEI 61131-3 est le même que le type Integer64.

### 5.3.1.7.4 Types non signés (Unsigned)

#### 5.3.1.7.4.1 Unsigned8

**CLASS:** **Data type**

**ATTRIBUTES:**

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

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

#### 5.3.1.7.4.2 USINT

Ce type de la CEI 61131-3 est le même que le type Unsigned8.

#### 5.3.1.7.4.3 unsigned char

Ce type de données est le même que le type Unsigned8.

#### 5.3.1.7.4.4 Unsigned16

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

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

#### 5.3.1.7.4.5 UINT

Ce type de la CEI 61131-3 est le même que le type Unsigned16.

#### 5.3.1.7.4.6 unsigned short

Ce type de données est le même que le type Unsigned16.

#### 5.3.1.7.4.7 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 |

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

#### 5.3.1.7.4.8 UDINT

Ce type de la CEI 61131-3 est le même que le type Unsigned32.

#### 5.3.1.7.4.9 unsigned long

Ce type de données est le même que le type Unsigned32.

#### 5.3.1.7.4.10 Unsigned64

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 56 |

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | Unsigned64 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

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

#### 5.3.1.7.4.11 ULINT

Ce type de la CEI 61131-3 est le même que le type Unsigned64.

### 5.3.1.8 Types de caractères OctetString

#### 5.3.1.8.1 OctetString1

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 30 |
| 2 | Data type Name | = | OctetString1 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 1 |

Ce type a une longueur égale à un octet.

#### 5.3.1.8.2 OctetString2

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 31 |
| 2 | Data type Name | = | OctetString2 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type a une longueur de deux octets.

#### 5.3.1.8.3 OctetString4

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 32 |
| 2 | Data type Name | = | OctetString4 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type a une longueur de quatre octets.

#### 5.3.1.8.4 OctetString8

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 33 |
| 2 | Data type Name | = | OctetString8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

Ce type octet string (chaîne d'octets) a une longueur de huit octets.

#### 5.3.1.8.5 OctetString16

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 34 |
| 2 | Data type Name | = | OctetString16 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 16 |

Ce type a une longueur de 16 octets.

#### 5.3.1.8.6 UUID

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | UUID |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2.1 | Field Name | = | Data1 |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | Data2 |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | Data3 |
| 5.2.6 | Field Data type | = | unsigned short |
| 5.2.7 | Field Name | = | Data4 |
| 5.2.8.1 | Format | = | ARRAY |
| 5.2.8.4.1 | Number of Array Elements | = | 8 |
| 5.2.8.4.2 | Array Element Data type | = | unsigned char |

Ce data type définit un type de données de longueur fixe égale à 16 octets. La sémantique est spécifiée par le modèle ORPC utilisé et ne relève pas du domaine d'application de la présente norme.

Ce type de données est structuré comme suit.

**Data1**
Ce champ contient les huit premiers chiffres hexadécimaux de l'UUID.

**Data2**
Ce champ contient le premier groupe de quatre chiffres hexadécimaux de l'UUID.

**Data3**
Ce champ contient le deuxième groupe de quatre chiffres hexadécimaux de l'UUID.

**Data4**
Ce champ contient une matrice de huit éléments. Les deux premiers éléments contiennent le troisième groupe de quatre chiffres hexadécimaux de l'UUID. Les six éléments restants contiennent les 12 chiffres hexadécimaux finals de l'UUID.

Des valeurs prédéfinies pour éléments de Type 10 sont montrées dans le Tableau 8.

**Tableau 8 – UUID**

| Valeur | Description |
|---|---|
| UUID_NULL | |
| UUID_IUnknown | Identifie de manière univoque l'interface IUnknown. |
| UUID_IDispatch | Identifie de manière univoque l'interface IDispatch. |
| UUID_ICBAPhysicalDevice | Identifie de manière univoque l'interface ICBAPhysicalDevice. |
| UUID_ICBABrowse | Identifie de manière univoque l'interface ICBABrowse. |
| UUID_ICBAPersist | Identifie de manière univoque l'interface ICBAPersist. |
| UUID_ICBALogicalDevice | Identifie de manière univoque l'interface ICBALogicalDevice. |

| Valeur | Description |
|---|---|
| UUID_ICBAState | Identifie de manière univoque l'interface ICBAState. |
| UUID_ICBATime | Identifie de manière univoque l'interface ICBATime. |
| UUID_ICBAGroupError | Identifie de manière univoque l'interface ICBAGroupError. |
| UUID_ICBAAccoMgt | Identifie de manière univoque l'interface ICBAAccoMgt. |
| UUID_ICBAAccoServer | Identifie de manière univoque l'interface ICBAAccoServer. |
| UUID_ICBAAccoCallback | Identifie de manière univoque l'interface ICBAAccoCallback. |
| UUID_ICBAAccoSync | Identifie de manière univoque l'interface ICBAAccoSync. |
| UUID_ICBARTAuto | Identifie de manière univoque l'interface ICBARTAuto. |
| UUID_PhysicalDevice | Identifie de manière univoque la classe Physical Device. |
| UUID_LogicalDevice | Identifie de manière univoque la classe Logical Device (appareil logique). |
| UUID_ACCO | Identifie de manière univoque la classe ACCO. |
| UUID_RTAuto | Identifie de manière univoque la classe RTAuto. |

#### 5.3.1.9 Types Pointer (pointeurs)

##### 5.3.1.9.1 Interface pointer (pointeur d'interface)

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | Interface Pointer |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type de données définit un type de données de longueur fixe égale à 4 octets.

##### 5.3.1.9.2 LPWSTR

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | LPWSTR |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type de données définit une référence à un UnicodeString.

#### 5.3.1.10 Types Time (temps)

##### 5.3.1.10.1 BinaryTime0

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 40 |
| 2 | Data type Name | = | BinaryTime0 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de deux octets. L'unité de temps pour ce type est de 10 µs.

##### 5.3.1.10.2 BinaryTime1

**CLASS:** **Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 41 |
| 2 | Data type Name | = | BinaryTime1 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de deux octets. L'unité de temps pour ce type est de 100 μs.

#### 5.3.1.10.3 BinaryTime2

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 42 |
| 2 | Data type Name | = | BinaryTime2 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de deux octets. L'unité de temps pour ce type est de 1 ms.

#### 5.3.1.10.4 BinaryTime3

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 43 |
| 2 | Data type Name | = | BinaryTime3 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de deux octets. L'unité de temps pour ce type est de 10 ms.

#### 5.3.1.10.5 BinaryTime4

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 44 |
| 2 | Data type Name | = | BinaryTime4 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de quatre octets. L'unité de temps pour ce type est de 10 μs.

#### 5.3.1.10.6 BinaryTime5

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 45 |
| 2 | Data type Name | = | BinaryTime5 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de quatre octets. L'unité de temps pour ce type est de 100 μs.

#### 5.3.1.10.7 BinaryTime6

| **CLASS:** | **Data type** |
|---|---|
| **ATTRIBUTES:** | |

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 46 |
| 2 | Data type Name | = | BinaryTime6 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de quatre octets. L'unité de temps pour ce type est de 1 ms.

#### 5.3.1.10.8 BinaryTime7

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 47 |
| 2 | Data type Name | = | BinaryTime7 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de six octets. L'unité de temps pour ce type est de 1 ms.

#### 5.3.1.10.9 BinaryTime8

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 48 |
| 2 | Data type Name | = | BinaryTime8 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 6 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur de six octets. L'unité de temps pour ce type est de 10 µs.

#### 5.3.1.10.10 BinaryTime9

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 49 |
| 2 | Data type Name | = | BinaryTime9 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 6 |

Ce type est un nombre binaire. Le bit de poids fort de l'octet de poids fort est toujours utilisé comme le bit de poids fort du nombre binaire; aucun bit de signe n'est inclus. Ce type binary time (temps binaire) a une longueur de six octets. L'unité de temps pour ce type est de 100 µs.

#### 5.3.1.10.11 TIME

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | TIME |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type de la CEI 61131-3 est un nombre binaire en complément à deux avec une longueur égale à quatre octets. L'unité de temps pour ce type est de 1 ms.

#### 5.3.1.10.12 ITIME

| CLASS: | | | Data type |
|---|---|---|---|
| ATTRIBUTES: | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | ITIME |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Cette extension de type de la CEI 61131-3 est un nombre binaire en complément à deux avec une longueur égale à deux octets. L'unité de temps pour ce type est de 1 ms.

#### 5.3.1.10.13 FTIME

| CLASS: | | | Data type |
|---|---|---|---|
| ATTRIBUTES: | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | FTIME |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Cette extension de type de la CEI 61131-3 est un nombre binaire en complément à deux avec une longueur égale à quatre octets. L'unité de temps pour ce type est de 1 µs.

#### 5.3.1.10.14 LTIME

| CLASS: | | | Data type |
|---|---|---|---|
| ATTRIBUTES: | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | LTIME |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

Cette extension de type de la CEI 61131-3 est un nombre binaire en complément à deux avec une longueur égale à huit octets. L'unité de temps pour ce type est de 1 µs.

#### 5.3.1.10.15 NetworkTime

| CLASS: | | | Data type |
|---|---|---|---|
| ATTRIBUTES: | | | |
| 1 | Data type Numeric Identifier | = | 58 |
| 2 | Data type Name | = | NetworkTime |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

Ce data type est constitué d'une valeur entière (Integer) et d'une valeur non signée (unsigned) qui expriment le temps réseau.

Le premier élément est un data type Unsigned32 qui donne le temps réseau en secondes à partir du 1900.01.01 (1$^{er}$ janvier 1900) 00:00:00(TUC) pour le temps réseau supérieur ou égal à 1984.01.01 (1$^{er}$ janvier 1984) 00:00:00 (TUC) et inférieur à 2036.07.02 (2 juillet 2036) 06:28:16(TUC), ou en secondes à partir du 2036.07.02 (2 juillet 2036) 06:28:16(TUC) pour le temps réseau supérieur ou égal à 2036.07.02 (2 juillet 2036) 06:28:16(TUC).

Le second élément est un data type Unsigned32 qui donne la partie fractionnaire de secondes en 1/2$^{32}$ s).

#### 5.3.1.10.16 NetworkTimeDifference

| CLASS: | | | Data type |
|---|---|---|---|
| ATTRIBUTES: | | | |
| 1 | Data type Numeric Identifier | = | 59 |
| 2 | Data type Name | = | NetworkTimeDifference |

| | | | |
|---|---|---|---|
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 8 |

Ce type de données est constitué d'une valeur entière (Integer) et d'une valeur non signée (unsigned) qui expriment la différence en temps réseau. Le premier élément est un type de données Integer32 qui fournit la différence de temps réseau en secondes. Le second élément est un type de données Unsigned32 qui donne la partie fractionnaire de secondes en $1/2^{32}$ s).

#### 5.3.1.11 Types de caractères VisibleString

##### 5.3.1.11.1 UNICODE char

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 36 |
| 2 | Data type Name | = | UnicodeChar |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type est défini comme un seul caractère dans le type chaîne UNICODE.

##### 5.3.1.11.2 VisibleString1

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 25 |
| 2 | Data type Name | = | VisibleString1 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 1 |

Ce type est défini comme un seul caractère dans le type VisibleString ISO.

##### 5.3.1.11.3 VisibleString2

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 26 |
| 2 | Data type Name | = | VisibleString2 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 2 |

Ce type contient deux éléments du type VisibleString.

##### 5.3.1.11.4 VisibleString4

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 27 |
| 2 | Data type Name | = | VisibleString4 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 4 |

Ce type contient quatre éléments du type VisibleString.

##### 5.3.1.11.5 VisibleString8

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | 28 |
| 2 | Data type Name | = | VisibleString8 |
| 3 | Format | = | FIXED LENGTH |

| | | | |
|---|---|---|---|
| 4.1 | Octet Length | = | 8 |

Ce type contient huit éléments du type VisibleString.

#### 5.3.1.11.6 VisibleString16

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 29 |
| 2 | Data type Name | = | VisibleString16 |
| 3 | Format | = | FIXED LENGTH |
| 4.1 | Octet Length | = | 16 |

Ce type contient 16 éléments du type VisibleString.

### 5.3.2 Types String (chaîne)

#### 5.3.2.1 BitString

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

Ce type chaîne est défini comme une série d'éléments BitString8.

#### 5.3.2.2 CompactBooleanArray

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 37 |
| 2 | Data type Name | = | CompactBooleanArray |
| 3 | Format | = | STRING |
| 6.1 | Octet Length | = | 1 |

Dans ce type, chaque valeur de bit égale à 0 (zéro) représente la valeur booléenne FALSE et chaque valeur de bit égale à 1 représente la valeur booléenne TRUE.

#### 5.3.2.3 CompactBCDArray

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 38 |
| 2 | Data type Name | = | CompactBCDArray |
| 3 | Format | = | STRING |
| 4.1 | Octet Length | = | 1 |

Ce type sert à compacter une série ordonnée de valeurs BCD, deux par octet, en une série ordonnée d'octets. Le premier octet contient la valeur BCD de poids fort dans le quartet de poids fort. Si le nombre de valeurs BCD est impair, le quartet de poids faible de l'octet final est mis à la valeur réservée "1111".

#### 5.3.2.4 OctetString

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 10 |
| 2 | Data type Name | = | OctetString |
| 3 | Format | = | STRING |

| | | | |
|---|---|---|---|
| 4.1 | Octet Length | = | 1 à n |

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 61158-6-5 définit l'ordre d'émission.

#### 5.3.2.5 UNICODEString

| | | | |
|---|---|---|---|
| **CLASS:** | | | **Data type** |
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | 39 |
| 2 | Data type Name | = | UnicodeString |
| 3 | Format | = | STRING |
| 4.1 | Octet Length | = | 2 |

Ce type est défini comme étant le type "string" (chaîne) UNICODE.

#### 5.3.2.6 VisibleString

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

Ce type est défini comme étant le type "string" (chaîne) ISO/CEI 646.

### 5.3.3 Types Structure

#### 5.3.3.1 ADDCONNECTIONIN

| | | | |
|---|---|---|---|
| **CLASS:** | **Data type** | | |
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | ADDCONNECTIONIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 5 |
| 5.2.1 | Field Name | = | ProviderItem |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | ConsumerItem |
| 5.2.4 | Field Data type | = | LPWSTR |
| 5.2.5 | Field Name | = | Persistence |
| 5.2.6 | Field Data type | = | PERSISTDEF |
| 5.2.7 | Field Name | = | SubstituteValue |
| 5.2.8 | Field Data type | = | VARIANT |
| 5.2.9 | Field Name | = | Epsilon |
| 5.2.10 | Field Data type | = | VARIANT |

Ce type de données définit le type de données ADDCONNECTIONIN.

Ce type de données est structuré comme suit.

**ProviderItem**
Ce champ contient le nom de l'élément de données source.

**ConsumerItem**
Ce champ contient le nom de l'élément de données puits.

**Persistence**
Ce champ décrit la persistance requise des informations de connexion. Les valeurs CBAVolatile et CBAPersistent sont autorisées.

**SubstituteValue**
Ce champ spécifie la valeur substitut selon le type de données de l'élément de connexion.

**Epsilon**
Ce champ spécifie l'hystérésis comme étant un changement absolu de la valeur.

#### 5.3.3.2 ADDCONNECTIONOUT

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | ADDCONNECTIONOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 3 |
| 5.2.1 | Field Name | = | ConsumerID |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | Version |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | ErrorState |
| 5.2.6 | Field Data type | = | HRESULT |

Ce type de données définit le type de données ADDCONNECTIONOUT.

Ce type de données est structuré comme suit.

**ConsumerID**
Ce champ contient l'identificateur de l'abonné.

**Version**
Ce champ contient le numéro de version de la connexion.

**ErrorState**
Ce champ contient la condition d'erreur de la connexion.

#### 5.3.3.3 BSTR

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | BSTR |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | OctetCount |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | UnicodeString |
| 5.2.4 | Field Data type | = | UnicodeString |

Ce type de données définit un UnicodeString avec une valeur de comptes d'octets qui précède. Le compte contient le nombre d'octets, pas celui de caractères UNICODE, dans la chaîne. Ce compte n'inclut pas le caractère zéro terminal (2 octets).

#### 5.3.3.4 CONNECTIN

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | CONNECTIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | ProviderItem |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | DataType |
| 5.2.4 | Field Data type | = | VARTYPE |
| 5.2.5 | Field Name | = | Epsilon |
| 5.2.6 | Field Data type | = | VARIANT |
| 5.2.7 | Field Name | = | ConsumerID |
| 5.2.8 | Field Data type | = | unsigned long |

Ce type de données définit le type de données CONNECTIN.

Ce type de données est structuré comme suit.

**ProviderItem**
Ce champ contient le nom de l'élément de données source.

**DataType**
Ce champ contient le code de type pour l'élément de donnée.

**Epsilon**
Ce champ spécifie l'hystérésis comme étant un changement absolu de la valeur.

**ConsumerID**
Ce champ contient l'identificateur de l'abonné.

#### 5.3.3.5 CONNECTOUT

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | CONNECTOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | ProviderID |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | ErrorState |
| 5.2.4 | Field Data type | = | HRESULT |

Ce type de données définit le type de données CONNECTOUT.

Ce type de données est structuré comme suit.

**ProviderID**
Ce champ contient l'identificateur du fournisseur.

**ErrorState**
Ce champ contient la condition d'erreur de l'interconnexion.

#### 5.3.3.6 EXCEPTINFO

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | EXCEPINFO |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 9 |
| 5.2.1 | Field Name | = | wCode |
| 5.2.2 | Field Data type | = | unsigned short |
| 5.2.3 | Field Name | = | wReserved |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | bstrSource |
| 5.2.6 | Field Data type | = | BSTR |
| 5.2.7 | Field Name | = | bstrDescription |
| 5.2.8 | Field Data type | = | BSTR |
| 5.2.9 | Field Name | = | bstrHelpFile |
| 5.2.10 | Field Data type | = | BSTR |
| 5.2.11 | Field Name | = | dwHelpContext |
| 5.2.12 | Field Data type | = | unsigned long |
| 5.2.13 | Field Name | = | pvReserved |
| 5.2.14 | Field Data type | = | unsigned long |
| 5.2.15 | Field Name | = | pfnDeferredFillIn |
| 5.2.16 | Field Data type | = | unsigned long |
| 5.2.17 | Field Name | = | scode |
| 5.2.18 | Field Data type | = | long |

Ce type de données définit le type de données EXCEPINFO pour décrire une exception qui se produit au cours du service Invoke.

Ce type de données est structuré comme suit.

**wCode**
Ce champ contient un code d'erreur identifiant une erreur. Il convient que les codes d'erreur soient supérieurs à 1000. Il convient que ce champ ou le champ code soit renseigné; l'autre étant positionné à 0.

**wReserved**
Ce champ est réservé et il convient de le mettre à 0.

**bstrSource**
Ce champ contient un nom textuel lisible par l'homme de la source de l'exception. Typiquement, il est un nom d'application. Il convient que ce champ soit renseigné par le réalisateur de l'interface IDispatch.

**bstrDescription**
Ce champ contient une description textuelle lisible par l'homme de l'erreur destinée au client. Si aucune description n'est disponible, utiliser Null.

**bstrHelpFile**
Ce champ contient l'ensemble qualifié de lecteur, chemin et nom de fichier d'un fichier d'aide (Help) avec plus d'informations relatives à l'erreur. Si aucun fichier d'aide n'est disponible, utiliser Null.

**dwHelpContext**
Ce champ contient l'identificateur de contexte Help relatif à la rubrique dans le fichier Help. Il convient que ce champ soit renseigné si et seulement si le champ bstrHelpFile n'est pas mis à Null.

**pvReserved**
Ce champ est réservé et il convient de le mettre à Null.

**pfnDeferredFillIn**
Ce champ contient l'adresse d'une fonction qui prend une structure EXCEPINFO comme argument et retourne une valeur HRESULT. Si différé, le remplissage n'est pas souhaité, il convient que ce champ soit mis à Null.

**scode**
Ce champ contient une valeur de retour décrivant l'erreur. Il convient de renseigner soit ce champ soit le champ wCode; l'autre étant positionné à 0.

#### 5.3.3.7 DATE_AND_TIME

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | DATE_AND_TIME |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Time_Of_Day_Element |
| 5.2.2 | Field Data type | = | TIME_OF_DAY |
| 5.3.1 | Field Name | = | Date_Element |
| 5.3.2 | Field Data type | = | DATE |

Cette extension de type de la CEI 61131-3 est une structure qui exprime tant la date (comme un nombre de jours commençant à partir du 1er janvier 1972 jusqu'au 6 juin 2151) que l'heure du jour comme un nombre de ms commençant à minuit.

#### 5.3.3.8 FILETIME

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | FILETIME |
| 3 | Format | = | STRUCTURE |

| | | | |
|---|---|---|---|
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | LowDataTime |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | HighDataTime |
| 5.2.4 | Field Data type | = | unsigned long |

Ce type de données est une valeur temporelle de 64 bits représentant le nombre d'intervalles de 100 nanosecondes depuis le 1er janvier 1601.

#### 5.3.3.9 GETIDOUT

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | GETIDOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2.1 | Field Name | = | ConsumerID |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | State |
| 5.2.4 | Field Data type | = | VT_BOOLEAN |
| 5.2.5 | Field Name | = | Version |
| 5.2.6 | Field Data type | = | unsigned short |
| 5.2.7 | Field Name | = | ErrorState |
| 5.2.8 | Field Data type | = | HRESULT |

Ce type de données définit le type de données GETIDOUT.

Ce type de données est structuré comme suit.

**ConsumerID**
Ce champ contient l'identificateur de l'abonné.

**State**
Ce champ contient l'état de la connexion. La valeur TRUE Indique une connexion active et la valeur FALSE une connexion inactive.

**Version**
Ce champ contient le numéro de version de la connexion.

**ErrorState**
Ce champ contient la condition d'erreur de la connexion.

#### 5.3.3.10 GETCONNECTIONOUT

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | GETCONNECTIONOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 10 |
| 5.2.1 | Field Name | = | Provider |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | ProviderItem |
| 5.2.4 | Field Data type | = | LPWSTR |
| 5.2.5 | Field Name | = | ConsumerItem |
| 5.2.6 | Field Data type | = | LPWSTR |
| 5.2.7 | Field Name | = | SubstituteValue |
| 5.2.8 | Field Data type | = | VARIANT |
| 5.2.9 | Field Name | = | Epsilon |
| 5.2.10 | Field Data type | = | VARIANT |
| 5.2.11 | Field Name | = | QoSType |
| 5.2.12 | Field Data type | = | unsigned short |
| 5.2.13 | Field Name | = | QoSValue |
| 5.2.14 | Field Data type | = | unsigned short |
| 5.2.15 | Field Name | = | State |
| 5.2.16 | Field Data type | = | VT_BOOLEAN |
| 5.2.17 | Field Name | = | Persistence |
| 5.2.18 | Field Data type | = | PERSISTDEF |

| | | | |
|---|---|---|---|
| 5.2.19 | Field Name | = | Version |
| 5.2.20 | Field Data type | = | unsigned short |
| 5.2.21 | Field Name | = | ErrorState |
| 5.2.22 | Field Data type | = | HRESULT |

Ce type de données définit le type de données GETCONNECTIONOUT.

Ce type de données est structuré comme suit.

**Provider**
Ce champ contient le nom de LDev du fournisseur (Format: PDev!LDev).

**ProviderItem**
Ce champ contient le nom de l'élément de données source.

**ConsumerItem**
Ce champ contient le nom de l'élément de données puits.

**SubstituteValue**
Ce champ spécifie la valeur substitut selon le type de données de l'élément de connexion.

**Epsilon**
Ce champ spécifie l'hystérésis comme étant un changement absolu de la valeur.

**QoSType**
Ce champ contient le type de qualité de service.

**QoSValue**
Ce champ contient le qualificateur de qualité de service.

**State**
Ce champ contient l'état de la connexion. La valeur TRUE Indique une connexion active et la valeur FALSE une connexion inactive.

**Persistence**
Ce champ décrit la persistance requise des informations de connexion. Les valeurs CBAVolatile et CBAPersistent sont autorisées.

**Version**
Ce champ contient le numéro de version de la connexion.

**ErrorState**
Ce champ contient la condition d'erreur de la connexion.

#### 5.3.3.11 QualifiedOctetString2

| | | | |
|---|---|---|---|
| **CLASS:** | **Data type** | | |
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | QualifiedOctetString2 |
| 5 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Value |
| 5.2.2 | Field Data type | = | Octet String(2) |
| 5.2.3 | Field Name | = | Status |
| 5.2.4 | Field Data type | = | Unsigned8 |

Ce type de données définit le QualifiedOctetString2.

Ce type de données est structuré comme suit.

**Value**
Ce champ contient la valeur en Octet String avec la longueur de 2.

**Status**
Ce champ décrit le statut de la valeur comme qualificateur.

#### 5.3.3.12 QualifiedFloat32

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | QualifiedFloat32 |
| 5 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Value |
| 5.2.2 | Field Data type | = | Float32 |
| 5.2.3 | Field Name | = | Status |
| 5.2.4 | Field Data type | = | Unsigned8 |

Ce type de données définit le QualifiedFloat32.

Ce type de données est structuré comme suit.

**Value**
Ce champ contient la valeur en Float32.

**Status**
Ce champ décrit le statut de la valeur comme qualificateur.

#### 5.3.3.13 QualifiedUnsigned8

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | QualifiedUnsigned8 |
| 5 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Value |
| 5.2.2 | Field Data type | = | Unsigned8 |
| 5.2.3 | Field Name | = | Status |
| 5.2.4 | Field Data type | = | Unsigned8 |

Ce type de données définit le QualifiedUnsigned8.

Ce type de données est structuré comme suit.

**Value**
Ce champ contient la valeur en Unsigned8.

**Status**
Ce champ décrit le statut de la valeur comme qualificateur.

#### 5.3.3.14 READITEMOUT

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | READITEMOUT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 4 |
| 5.2.1 | Field Name | = | Value |
| 5.3.2 | Field Data type | = | VARIANT |
| 5.4.3 | Field Name | = | QualityCode |
| 5.5.4 | Field Data type | = | ITEMQUALITYDEF |
| 5.6.5 | Field Name | = | TimeStamp |
| 5.7.6 | Field Data type | = | FILETIME |
| 5.8.7 | Field Name | = | ErrorState |
| 5.9.8 | Field Data type | = | HRESULT |

Ce type de données définit le type de données READITEMOUT.

Ce type de données est structuré comme suit.

**Value**
Ce champ contient la valeur elle-même avec le format conforme au type de données approprié.

**QualityCode**
Ce champ contient le Quality Code (code Quality) de la valeur de données.

**TimeStamp**
Ce champ contient le marqueur temporel de la valeur.

**ErrorState**
Ce champ contient la condition d'erreur de la connexion.

#### 5.3.3.15 SAFEARRAY

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | RGSABOUND |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Elements |
| 5.2.2 | Field Data type | = | unsigned long |
| 5.2.3 | Field Name | = | Left Bound |
| 5.2.4 | Field Data type | = | long |

Ce type de données est une aide décrivant un élément de matrice avec des informations relatives aux frontières nécessaires pour définir la matrice SAFEARRAY en dessous.

| **CLASS:** | **Data type** | | |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 2 | Data type Name | = | SAFEARRAY |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 6 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Dims |
| 5.2.2 | Field Data type | = | unsigned short |
| 5.2.3 | Field Name | = | Features |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | Elements |
| 5.2.6 | Field Data type | = | unsigned long |
| 5.2.7 | Field Name | = | Locks |
| 5.2.8 | Field Data type | = | unsigned long |
| 5.2.9 | Field Name | = | Data Address |
| 5.2.10 | Field Data type | = | unsigned long |
| 5.2.11 | Field Name | = | Rgsabound |
| 5.2.12 | Format | = | ARRAY |
| 5.2.12.1 | Number of Array Elements | = | Dims |
| 5.2.12.2 | Array Element Data type | = | RGSABOUND |

Ce type de données exprime une matrice unidimensionnelle ou multidimensionnelle d'un seul type de données. (Cependant, ce seul type de données peut être un VARIANT, vous accordant des matrices de différents types.) Ces matrices sont appelées "safe array" ("matrices sûres") parce qu'elles contiennent des informations relatives aux bornes, permettant de vérifier l'indice ou les indices par rapport aux bornes avant d'accéder aux données de la matrice. La borne inférieure de la matrice peut ne pas être zéro et, donc, il faut que la matrice sûre stocke sa borne inférieure ainsi que sa taille. Enfin, les matrices sûres permettent le verrouillage (et le déverrouillage) afin de garantir que le pointeur vers les données que vous obtenez est valide.

Ce type de données est structuré comme suit.

**Dims**
Ce champ contient la dimension de la matrice SAFEARRAY.

**Features**
Ce champ contient des fanions pour le type d'allocation et le type de données de la matrice SAFEARRAY.

**Elements**
Ce champ contient la taille d'un élément simple de la matrice SAFEARRAY.

**Locks**
Ce champ contient le compteur de verrouillages de la matrice SAFEARRAY.

**Data Address**
Ce champ contient l'adresse des données effectives de la matrice SAFEARRAY.

**Rgsabound**
Ce champ contient une matrice avec les informations relatives aux bornes pour chaque dimension de la matrice SAFEARRAY. Par conséquent, le nombre d'éléments de matrice est fonction de la dimension de la matrice sûre.

#### 5.3.3.16 SHORT_STRING

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | SHORT_STRING |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Charcount_Element |
| 5.2.2 | Field Data type | = | USINT |
| 5.3.1 | Field Name | = | Stringcontents_Element |
| 5.3.2 | Field Data type | = | OctetString |

Cette extension de type de la CEI 61131-3 est constituée de deux éléments. Charcount_Element donne le nombre courant de caractères dans Stringcontents_Element (un USINT par caractère).

#### 5.3.3.17 STRING

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | STRING |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Charcount_Element |
| 5.2.2 | Field Data type | = | UINT |
| 5.3.1 | Field Name | = | Stringcontents_Element |
| 5.3.2 | Field Data type | = | OctetString |

Ce type de la CEI 61131-3 est constitué de deux éléments. Charcount_Element donne le nombre courant de caractères dans Stringcontents_Element (un USINT par caractère).

#### 5.3.3.18 STRING2

| **CLASS:** | | | **Data type** |
|---|---|---|---|
| **ATTRIBUTES:** | | | |
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | STRING2 |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |

| | | | |
|---|---|---|---|
| 5.2.1 | Field Name | = | Charcount_Element |
| 5.2.2 | Field Data type | = | UINT |
| 5.3.1 | Field Name | = | String2contents_Element |
| 5.3.2 | Field Data type | = | OctetString |

Cette extension de type de la CEI 61131-3 est constituée de deux éléments. Charcount_Element donne le nombre courant de caractères dans String2contents_Element (un UINT par caractère).

#### 5.3.3.19 STRINGN

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | Data type Numeric Identifier | = | non utilisé |
| 2 | Data type Name | = | STRINGN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 3 |
| 5.2.1 | Field Name | = | Charsize_Element |
| 5.2.2 | Field Data type | = | UINT |
| 5.3.1 | Field Name | = | Charcount_Element |
| 5.3.2 | Field Data type | = | UINT |
| 5.4.1 | Field Name | = | StringNcontents_Element |
| 5.4.2 | Field Data type | = | OctetString |

Cette extension de type de la CEI 61131-3 est constituée de trois éléments. Charsize_Element donne la taille d'un caractère dans StringNcontents_Element (N = nombre de USINT). Charcount_Element donne le nombre courant de caractères dans StringNcontents_Element (N USINT par caractère).

#### 5.3.3.20 VARIANT

**CLASS:** **Data type**

**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | VARIANT |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 5 |
| 5.2 | List of Fields | | |
| 5.2.1 | Field Name | = | Tag |
| 5.2.2 | Field Data type | = | VARTYPE |
| 5.2.3 | Field Name | = | Padding1 |
| 5.2.4 | Field Data type | = | unsigned short |
| 5.2.5 | Field Name | = | Padding2 |
| 5.2.6 | Field Data type | = | unsigned short |
| 5.2.7 | Field Name | = | Padding3 |
| 5.2.8 | Field Data type | = | unsigned short |
| 5.2.9 | Field Name | = | Value |
| 5.2.10 | Field Data type | = | voir Tableau 9 |

Ce type de données exprime un type VARIANT contenant des valeurs possibles des types de données selon le Tableau 9. Le champ Tag contient l'identificateur numérique du type de données pour identifier le type de données de la valeur. La taille globale du type VARIANT est 16 octets.

**Tableau 9 – Noms de data types pour la valeur**

| Data type |
|---|
| VT_BOOLEAN |
| char |
| short |
| long |

| Data type |
|---|
| unsigned char |
| unsigned short |
| unsigned long |
| float |
| double |
| date |
| currency |
| adresse d'un BSTR |
| adresse d'une matrice SAFEARRAY |
| adresse d'une userdefined struct (structure définie par l'utilisateur) |
| Interface Pointer (Pointeur d'interface) |
| HRESULT |

#### 5.3.3.21 WRITEITEMIN

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | WRITEITEMIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 2 |
| 5.2.1 | Field Name | = | Item |
| 5.2.2 | Field Data type | = | LPWSTR |
| 5.2.3 | Field Name | = | Value |
| 5.2.4 | Field Data type | = | VARIANT |

Ce type de données définit le type de données WRITEITEMIN.

Ce type de données est structuré comme suit.

**Item**
Ce champ contient le nom de l'élément de données.

**Value**
Ce champ contient la valeur elle-même avec le format conforme au type de données approprié.

#### 5.3.3.22 WRITEITEMQCDIN

**CLASS: Data type**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 2 | Data type Name | = | WRITEITEMQCDIN |
| 3 | Format | = | STRUCTURE |
| 5.1 | Number of Fields | = | 3 |
| 5.2.1 | Field Name | = | WriteItem |
| 5.2.2 | Field Data type | = | WRITEITEMIN |
| 5.2.3 | Field Name | = | QualityCode |
| 5.2.4 | Field Data type | = | ITEMQUALITYDEF |
| 5.2.5 | Field Name | = | TimeStamp |
| 5.2.6 | Field Data type | = | FILETIME |

Ce type de données définit le type de données WRITEITEMQCDIN.

Ce type de données est structuré comme suit.

**WriteItem**
Ce champ contient le nom et la valeur de l'élément de données.

**QualityCode**
Ce champ contient le Quality Code (code Quality) de la valeur de données.

**TimeStamp**
Ce champ contient le marqueur temporel de la valeur.

### 5.3.3.23 DISPPARAMS

| | | |
|---|---|---|
| 2 | Data type Name = | UUID |
| 3 | Format = | STRUCTURE |
| 5.1 | Number of Fields = | 4 |
| 5.2.1 | Field Name = | Data1 |
| 5.2.2 | Field Data type = | unsigned long |
| 5.2.3 | Field Name = | Data2 |
| 5.2.4 | Field Data type = | unsigned short |
| 5.2.5 | Field Name = | Data3 |
| 5.2.6 | Field Data type = | unsigned short |
| 5.2.7 | Field Name = | Data4 |
| 5.2.8.1 | Format = | ARRAY |
| 5.2.8.4.1 | Number of Array Elements = | 8 |
| 5.2.8.4.2 | Array Element Data type = | unsigned char |

Ce type de données définit un type de données de longueur fixe égale à 16 octets. La sémantique est spécifiée par le modèle ORPC utilisé et ne relève pas du domaine d'application de la présente norme.

Ce type de données est structuré comme suit.

**Data1**
Ce champ contient les huit premiers chiffres hexadécimaux de l'UUID.

**Data2**
Ce champ contient le premier groupe de quatre chiffres hexadécimaux de l'UUID.

**Data3**
Ce champ contient le deuxième groupe de quatre chiffres hexadécimaux de l'UUID.

**Data4**
Ce champ contient une matrice de huit éléments. Les deux premiers éléments contiennent le troisième groupe de quatre chiffres hexadécimaux de l'UUID. Les six éléments restants contiennent les 12 chiffres hexadécimaux finals de l'UUID.

Des valeurs prédéfinies pour éléments de Type 10 sont montrées dans le Tableau 10.

**Tableau 10 – UUID**

| Value | Description |
|---|---|
| UUID_NULL | |
| UUID_IUnknown | Identifie de manière univoque l'interface IUnknown. |
| UUID_IDispatch | Identifie de manière univoque l'interface IDispatch. |
| UUID_ICBAPhysicalDevice | Identifie de manière univoque l'interface ICBAPhysicalDevice. |
| UUID_ICBABrowse | Identifie de manière univoque l'interface ICBABrowse. |
| UUID_ICBAPersist | Identifie de manière univoque l'interface ICBAPersist. |
| UUID_ICBALogicalDevice | Identifie de manière univoque l'interface ICBALogicalDevice. |
| UUID_ICBAState | Identifie de manière univoque l'interface ICBAState. |
| UUID_ICBATime | Identifie de manière univoque l'interface ICBATime. |
| UUID_ICBAGroupError | Identifie de manière univoque l'interface ICBAGroupError. |
| UUID_ICBAAccoMgt | Identifie de manière univoque l'interface ICBAAccoMgt. |
| UUID_ICBAAccoServer | Identifie de manière univoque l'interface ICBAAccoServer. |
| UUID_ICBAAccoCallback | Identifie de manière univoque l'interface ICBAAccoCallback. |
| UUID_ICBAAccoSync | Identifie de manière univoque l'interface ICBAAccoSync. |

| Value | Description |
|---|---|
| UUID_ICBARTAuto | Identifie de manière univoque l'interface ICBARTAuto. |
| UUID_PhysicalDevice | Identifie de manière univoque la classe Physical Device. |
| UUID_LogicalDevice | Identifie de manière univoque la classe Logical Device (appareil logique). |
| UUID_ACCO | Identifie de manière univoque la classe ACCO. |
| UUID_RTAuto | Identifie de manière univoque la classe RTAuto. |

### 5.4 Spécification du service Data type ASE

Il n'y a pas de services opérationnels définis pour l'objet type.

## 6 Spécification de modèle de communication

### 6.1 Concepts

Les concepts sont ceux du modèle de communications de base décrit dans la CEI 61158-1.

### 6.2 ASE

#### 6.2.1 ASE Object management

##### 6.2.1.1 Vue d'ensemble

Les modèles d'ASE de la FAL spécifient chacun une ou plusieurs classes APO de FAL connexes comme un ensemble d'attributs et de services. Les services de la FAL définis pour une classe sont utilisés pour manipuler, ou commander des APO de la classe ou accéder à ceux-ci. Les services qui sont pris en charge par un APO spécifique sont spécifiés par l'attribut ListOfManagementServices de l'APO.

Pour créer ou supprimer de façon dynamique la vue réseau d'un APO, ou pour accéder à ses valeurs d'attributs, cet ASE de la FAL définit des services communs de gestion d'APO de la FAL. Chaque définition de classe APO spécifie lequel de ces services peut être inclus dans les définitions des APO de la classe. Ces services fonctionnent en utilisant le modèle client/serveur.

##### 6.2.1.2 Spécification du modèle de gestion de la FAL

Les services de gestion spécifiés par le Modèle de gestion de la FAL opèrent sur les APO de la FAL et leurs attributs. Les classes FAL APO sont définies au sein des ASE de la FAL qui sont spécifiés dans les articles restants de la norme. Chaque classe FAL APO définie est décrite par un jeu d'attributs et un jeu de services.

##### 6.2.1.3 Services du modèle de gestion de la FAL

###### 6.2.1.3.1 Services pris en charge

Les services de gestion sont définis pour gérer des AP et des APO. Pour la simplicité, les AP et les APO sont appelés objets dans le reste de 6.2.1.3.

Les services Create et Delete sont spécifiés pour une classe si la vue réseau d'un objet de la classe peut être créée ou supprimée de façon dynamique. Les services Get Attributes et Set Attributes services sont spécifiés pour indiquer le moment où ses attributs peuvent être écrits ou lus. Les attributs d'un APO qui peuvent être réglés sont spécifiés comme partie intégrante de la définition de l'attribut.

Parce que des changements imprévus des définitions de l'objet au sein d'un AP peuvent avoir une incidence sur le fonctionnement de l'AP, les services Begin Set Attributes et End Set Attributes sont définis. Le service Begin Set Attributes sert à demander à l'AP distant la permission de modifier ses définitions d'objets. Le service End Set Attributes sert à indiquer à l'AP distant que les changements apportés à ses définitions d'objets sont achevés et qu'il peut reprendre le fonctionnement normal.

#### 6.2.1.3.2 Service Create

##### 6.2.1.3.2.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour rendre un objet visible et accessible à travers le réseau. Comme partie intégrante du service, des valeurs initiales peuvent être spécifiées pour les attributs d'objet. Si tous les attributs devant être initialisés ne peuvent être acheminés dans une seule APDU, le service Set Attributes peut être utilisé une fois que l'objet a été créé pour les attributs pouvant être mis à jour à l'aide du service Set Attributes. L'étendue de la visibilité pour les objets créés est spécifique à l'Application Process (processus application).

Le paramètre List of Supported Attributes sert à indiquer au demandeur les attributs qui ont été réellement créés pour l'objet. Si des valeurs initiales ont été fournies pour les attributs qui n'avaient pas été créés, le paramètre List of Supported Attributes peut être utilisé pour déterminer ceux qui n'avaient été créés et initialisés conformément à la demande. Cette capacité représente une forme de négociation lors de la création d'un objet.

NOTE Le fait qu'un AP crée réellement ou non l'objet relève d'une initiative locale. L'intention du service est de demander qu'il soit préparé pour un accès distant.

##### 6.2.1.3.2.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 11.

**Tableau 11 – Paramètres du service Create**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID (Identificateur d'invocation) | U | | | |
| ASE de FAL/Classe de FAL | M | M (=) | | |
| List of Attribute IDs and Initial Values | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | M (=) |
| Numeric ID | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

**Argument**

L'argument contient les paramètres de la demande de service.

**List of Attribute IDs and Initial Values**

Ce paramètre identifie et spécifie la valeur initiale d'un ou plusieurs des attributs de l'objet devant être réglés au cours du processus de création. Il convient qu'au moins l'un des attributs-clés de l'objet soit présent. Des valeurs pour les attributs obligatoires et facultatifs peuvent être fournies. L'AP qui crée l'objet détermine le complément d'attributs pris en charge pour l'objet.

NOTE Les valeurs initiales sont seulement requises pour les attributs obligatoires si l'AP créant l'objet n'a pas la capacité de fournir une valeur initiale lui-même. La manière dont il acquiert ces valeurs ne relève pas du domaine d'application de la présente norme.

**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.

#### 6.2.1.3.2.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.1.3.3 Service Delete

#### 6.2.1.3.3.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour rendre invisible et inaccessible à travers le réseau un objet qui est actuellement visible et accessible.

NOTE Le fait qu'un objet réel soit effectivement supprimé ou simplement enlevé de l'accessibilité du réseau relève d'une initiative locale.

#### 6.2.1.3.3.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 12.

**Tableau 12 – Paramètres du service Delete**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| ASE de FAL/Classe de FAL | M | M (=) | | |
| Key Attribute | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande de service.

**FAL ASE/FAL Class**

Ce paramètre spécifie l'ASE de FAL (AP, AR, Variable, Data type, Event, Function invocation, Load Region) et la classe de FAL dans l'ASE (par exemple: AREP, Variable List, Notifier, Action).

**Key Attribute**

Ce paramètre identifie l'objet par l'un de ses attributs-clés.

**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.

#### 6.2.1.3.3.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Une fois supprimé, un objet n'est plus accessible par les services de la FAL.

### 6.2.1.3.4 Services Get attributes

#### 6.2.1.3.4.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour lire les valeurs courantes d'une liste d'attributs pour un ou plusieurs objets d'une ou plusieurs classes définies pour l'AP. Il opère dans un mode « best effort »best, signifiant que le service réussit si la valeur d'au moins un attribut demandé pour un objet demandé est retournée.

#### 6.2.1.3.4.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 13.

**Tableau 13 – Paramètre du service Get attributes**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List of Objects and their Attributes | S | S (=) | | |
| Key Attribute | M | M (=) | | |
| List of Requested Attributes | M | M (=) | | |
| Data type Requested | C | C (=) | | |
| Range of Objects and their Attributes | S | S (=) | | |
| Starting Numeric ID | M | M (=) | | |
| List of Requested Attributes | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| More | | | M | M (=) |
| List of Responses | | | M | M (=) |
| List of Supported Attributes | | | C | C (=) |
| Error Status | | | S | S (=) |
| List of Attribute Values | | | S | S (=) |
| Data type Description | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE voir la Note en 3.8.4.3.

**Argument**

L'argument contient les paramètres de la demande de service.

**List of Objects and their Attributes**

Ce paramètre de type sélection fournit une liste qui identifie chaque objet et sa classe ainsi que les attributs qui sont demandés. Trois valeurs pour identifier des attributs ont possibles, à savoir tous les attributs, les attributs obligatoires seulement et une liste d'attributs individuels. Il est présent si l'utilisateur demande des attributs pour une liste d'objets.

**Key Attribute**

Ce paramètre identifie l'objet pour lequel des attributs sont demandés à l'aide de l'un des attributs-clés de l'objet.

**List of Requested Attributes**

Ce paramètre identifie un ou plusieurs des attributs pour lesquels des valeurs sont demandées.

**Range of Objects and their Attributes**

Ce paramètre de type sélection fournit une plage d'identificateurs numériques pour des objets d'une seule classe, et facultativement pour une série d'objets de classes différentes, et les attributs qui sont demandés. Trois valeurs pour identifier des attributs ont possibles, à savoir tous les attributs, les attributs obligatoires seulement et une liste d'attributs individuels. Il est présent si l'utilisateur demande des attributs pour des objets au sein d'une plage d'identificateurs numériques.

**Starting Numeric ID**

Ce paramètre spécifie le premier dans une plage d'identificateurs numériques. Les attributs demandés sont pour tous les objets commençant par cet identificateur numérique et après celui-ci. La valeur ALL peut être utilisée pour indiquer que les attributs pour tous les objets pour la classe spécifiée sont à retourner.

NOTE Des demandes multiples peuvent devoir être émises avec les identificateurs numériques successivement plus élevés lorsque toutes les définitions d'objets souhaitées ne peuvent pas être retournées dans une seule réponse.

**List of Requested Attributes**

Ce paramètre identifie un ou plusieurs des attributs pour lesquels des valeurs sont demandées.

**Data type Requested**

Ce paramètre conditionnel est présent seulement lorsque les attributs d'un objet variable sont demandés. Il indique si, oui ou non, la description des types de données est à retourner avec les attributs variables demandés.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**More**

Ce paramètre Boolean indique, lorsqu'il est FALSE, que les réponses pour tous les attributs demandés sont retournées. Si la valeur est TRUE, seules les réponses pour un sous-ensemble correct des attributs sont retournées. Cette situation se produit seulement lorsque toutes les valeurs ne s'inscrivent pas dans une seule APDU de réponse. Lorsque cela se produit, il est nécessaire que l'utilisateur présente une demande complémentaire pour les attributs manquants. Des valeurs partielles pour les attributs ne sont jamais retournées.

**List of Responses**

Ce paramètre spécifie un indicateur de statut ou une liste de valeurs d'attributs pour chaque objet spécifié dans la demande. Les réponses dans la liste sont retournées dans le même

ordre dans lequel les objets sont spécifiés dans la demande. L'objet associé à chaque réponse est identifié de façon explicite seulement si la valeur de l'un des attributs-clés avait été demandée. Si toutes les réponses n'ont pas pu être retournées dans une seule APDU, le paramètre More est positionné.

**List of Supported Attributes**

Ce paramètre identifie les attributs pris en charge par le répondeur pour l'objet spécifié. Il est présent si l'objet spécifié existe.

**Error Status**

Ce paramètre est retourné si le service Get a échoué pour tous les attributs d'un objet. Il indique la cause la plus probable de l'échec de l'opération en utilisant la classe d'erreurs et le code d'erreur définis pour le service Get Attributes.

**List of Attribute Values**

Ce paramètre est retourné si le service Get a réussi pour n'importe lequel ou la totalité des attributs demandés. Ce paramètre spécifie une liste de valeurs d'attribut, chacune étant identifiée individuellement et complétée, pour l'objet demandé.

**Data type Description**

Ce paramètre conditionnel est présent seulement lorsque la description de types de données pour un objet variable a été demandée. Il contient la description complète des types de données pour la variable.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.1.3.4.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Get Attributes est basé sur le principe "best effort". "best effort" signifie que le service réussit si au moins une valeur est retournée. Si l'utilisateur est capable de récupérer une ou plusieurs des valeurs d'attributs spécifiées et si celles-ci peuvent être acheminées dans une seule APDU, elles sont retournées dans la primitive response (+) du service Get Attributes.

Si elles ne peuvent pas être acheminées dans une seule APDU, l'utilisateur retourne celles qui le peuvent et met le fanion More dans la primitive response (+) du service Get Attributes. Le demandeur a alors la responsabilité de présenter des demandes complémentaires identifiant les attributs qui restent à récupérer.

Si l'utilisateur est incapable de récupérer la valeur d'au moins un attribut pour un objet de la liste, le service échoue et l'utilisateur émet une primitive response (-) du service Get Attributes indiquant la cause.

### 6.2.1.3.5 Services Set attributes

#### 6.2.1.3.5.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour mettre à jour la valeur courante d'une liste d'attributs pour un ou plusieurs objets. Ce service opère en mode tout ou rien — si tous les attributs demandés pour tous les objets ne peuvent pas être renseignés, le service échoue.

NOTE Ce service peut être utilisé pour charger la totalité des définitions d'objets en indiquant que tous les attributs obligatoires sont à mettre à jour.

#### 6.2.1.3.5.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 14.

**Tableau 14 – Paramètres du service Set attributes**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List of Object Definitions | M | M (=) | | |
| FAL ASE / FAL Class | M | M (=) | | |
| Key Attribute | M | M (=) | | |
| List of Attribute Updates | M | M (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| List of Supported Attributes | | | C | C (=) |
| NOTE voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande de service.

**List of Object Updates**

Ce paramètre spécifie la liste des objets et leurs attributs qui sont à mettre à jour.

**FAL ASE/FAL Class**

Ce paramètre spécifie l'ASE de FAL (AP, AR, Variable, Data type, Event, Function invocation, Load Region) et la classe de FAL dans l'ASE (par exemple: AREP, Variable List, Notifier, Action).

**Key Attribute**

Ce paramètre identifie l'objet par l'un de ses attributs-clés.

**List of Attribute Updates**

Ce paramètre identifie un ou plusieurs des attributs qui sont à mettre à jour et leurs valeurs.

**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.

**List of Supported Attributes**

Ce paramètre conditionnel identifie les attributs qui sont pris en charge par l'objet. Il est présent lorsqu'une demande a été reçue pour mettre à jour des attributs non pris en charge.

#### 6.2.1.3.5.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Set Attributes est un service "tout ou rien". "Tout ou rien" signifie que le service réussit seulement si toutes les mises à jour demandées ont réussi.

### 6.2.1.3.6 Service Begin set attributes

#### 6.2.1.3.6.1 Vue d'ensemble du service

Le service Begin Set Attributes prépare l'AP aux mises à jour de ses définitions d'objets, afin que les mises à jour soient exemptes d'interaction ou non exemptes d'interaction. L'AP retourne une réponse de réussite pour indiquer au demandeur qu'il peut commencer à mettre à jour les définitions d'objets.

#### 6.2.1.3.6.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 15.

**Tableau 15 – Service Begin set attributes**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Consequence | M | M (=) | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |

**Argument**

L'argument contient les paramètres de la demande de service.

**Consequence**

Ce paramètre énonce si les modifications prévues sont sans interaction pour les autres partenaires de communication, interagissent avec des mises à jour des définitions d'objets individuelles ou interagissent avec un nouveau chargement complet de toutes les définitions d'objets.

- chargement sans interaction
- recharge, avec interaction
- nouvelle charge, sans interaction

**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.

#### 6.2.1.3.6.3 Procédure de service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.1.3.7 Service End set attributes

#### 6.2.1.3.7.1 Vue d'ensemble du service

Le service End Set Attributes met fin au processus de mise à jour des définitions d'objets. Il sert à indiquer à l'AP que toutes les mises à jour de définitions d'objets ont été achevées.

#### 6.2.1.3.7.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 16.

**Tableau 16 – Service End set attributes**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Numeric ID | | | C | C (=) |

**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.

**Numeric ID**

Ce paramètre donne le Numeric ID (identificateur numérique) de l'objet, pour lequel la génération n'a pas réussi.

#### 6.2.1.3.7.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

## 6.2.2 ASE de processus application

### 6.2.2.1 Vue d'ensemble

La présente norme modélise un Application Process (processus application (AP)) de bus de terrain. Les processus application de bus de terrain représentent les informations et les ressources de traitement d'un système qui peuvent être accessibles par le biais de services de la FAL.

L'élément de service application dans la FAL qui fournit ces services est appelé un ASE Application Process. Dans l'ASE d'AP, l'AP est modélisé et accessible comme un APO avec un identificateur normalisé et prédéfini.

#### 6.2.2.2 Spécification de classe d'AP

##### 6.2.2.2.1 Modèle formel

La classe d'AP spécifie les attributs et les services définis pour des processus application. Sa classe parent "top" indique le sommet de l'arbre de la classe FAL. L'identificateur numérique pour l'Objet AP est toujours 0. L'utilisation de cette valeur réservée permet d'utiliser des services de gestion pour des objets d'AP sans connaître leurs noms.

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE d'AP** |
| **CLASS:** | | | AP |
| **CLASS ID:** | | | 16 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTS DES SERVICES IDENTIFY, GET STATUS et STATUS NOTIFICATION:** | | | |
| 1 | (m) | Attribut: | List Of AP Service Attributes |
| 1.1 | (m) | Attribut: | Manufacturer Identifier |
| 1.1.1 | (m) | Attribut: | Vendor Name |
| 1.1.2 | (m) | Attribut: | Model Identifier |
| 1.1.3 | (m) | Attribut: | Vendor Revision |
| 1.1.4 | (o) | Attribut: | Serial Number (Numéro de série) |
| 1.2 | (o) | Attribut: | AP Descriptor Reference |
| 1.3 | (m) | Attribut: | Network Access State |
| 1.3.1 | (m) | Attribut: | Service Access Status |
| 1.3.2 | (m) | Attribut: | Operational Status |
| **ATTRIBUTS DE LA GESTION SYSTÈME:** | | | |
| 2 | (m) | Attribut: | List Of System Management Attributes |
| 2.1 | (m) | Attribut: | List Of AR Endpoint Info |
| 2.1.1 | (m) | Attribut: | Numeric Id |
| 2.1.2 | (m) | Attribut: | Configured Max FAL PDU Size Sending |
| 2.1.3 | (m) | Attribut: | Configured Max FAL PDU Size Receiving |
| 2.1.4 | (m) | Attribut: | Configured Max Outstanding Services Requesting |
| 2.1.5 | (m) | Attribut: | Configured Max Outstanding Services Responding |
| 2.1.6 | (m) | Attribut: | List of Supported Services |
| 2.1.7 | (o) | Attribut: | Configured Max Data Structure Nesting Level |
| 2.2 | (m) | Attribut: | List Of In Use AR Endpoint Info |
| 2.2.1 | (m) | Attribut: | Context State |
| 2.2.2 | (m) | Attribut: | Negotiated Max FAL PDU Size Sending |
| 2.2.3 | (m) | Attribut: | Negotiated Max FAL PDU Size Receiving |
| 2.2.4 | (m) | Attribut: | Negotiated Max Outstanding Services Requesting |
| 2.2.5 | (m) | Attribut: | Negotiated Max Outstanding Services Responding |
| 2.2.6 | (m) | Attribut: | Outstanding Services Requesting Counter |
| 2.2.7 | (m) | Attribut: | Outstanding Services Responding Counter |
| 2.2.8 | (o) | Attribut: | Negotiated Max Data Structure Nesting Level |
| 2.2.9 | (o) | Attribut: | Negotiated List of Supported Services |
| **ATTRIBUTS DE LA GESTION D'OBJETS** | | | |
| 3 | (m) | Attribut: | List Of Managed AP Attributes |
| 3.1 | (m) | Attribut: | List Of Supported APO Classes and Objects |
| 3.1.1 | (m) | Attribut: | List Header |
| 3.1.1.1 | (o) | Attribut: | Numeric Identifier = 0 |
| 3.1.1.2 | (m) | Attribut: | ROM / RAM Flag (TRUE, FALSE) |

| | | | |
|---|---|---|---|
| 3.1.1.3 | (m) | Attribut: | Max Name Length |
| 3.1.1.4 | (m) | Attribut: | Access Protection Supported (TRUE, FALSE) |
| 3.1.1.5 | (m) | Attribut: | Version Of List |
| 3.1.1.6 | (m) | Attribut: | List Local Reference |
| 3.1.2 | (c) | Constraint | Data type Supported |
| 3.1.2.1 | (o) | Attribut: | Numeric Identifier = 1 |
| 3.1.2.2 | (m) | Attribut: | Number Of Data type Objects |
| 3.1.2.3 | (m) | Attribut: | Local Reference |
| 3.1.3 | (c) | Constraint | Variable \|\| Load Region \|\| Event Supported |
| 3.1.3.1 | (o) | Attribut: | Static Object Numeric Identifier |
| 3.1.3.2 | (m) | Attribut: | Number Of Static Objects (Variables + Load Region + Events) |
| 3.1.3.3 | (m) | Attribut: | Static Object Local Reference |
| 3.1.4 | (c) | Constraint | Variable List Supported |
| 3.1.4.1 | (o) | Attribut: | Numeric Identifier |
| 3.1.4.2 | (m) | Attribut: | Number Of Variable List Objects |
| 3.1.4.3 | (m) | Attribut: | Variable List Local Reference |
| 3.1.5 | (c) | Constraint | Function Invocation Supported |
| 3.1.5.1 | (o) | Attribut: | Function Invocation Numeric Identifier |
| 3.1.5.2 | (m) | Attribut: | Number Of Function Invocation Objects |
| 3.1.5.3 | (m) | Attribut: | Function Invocation Local Reference |
| 3.1.6 | (c) | Constraint | Dynamic Load Region |
| 3.1.6.1 | (o) | Attribut: | Load Region Numeric Identifier |
| 3.1.6.2 | (m) | Attribut: | Number of Load Region Objects |
| 3.1.6.3 | (m) | Attribut: | Load Region Local Reference |
| 3.2 | (o) | Attribut: | List Of DLSAP Addresses |
| 3.3 | (o) | Attribut: | List of Event Summary Selection Criteria |

**SUPPORTED ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 5 | (o) | Attribut: | List Of Supported Attributes |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | Ops Service: | Subscribe |
| 2 | (o) | Ops Service: | Identify (c'est-à-dire identifier) |
| 3 | (o) | Ops Service: | Get Status |
| 4 | (o) | Ops Service: | Status Notification |
| 5 | (o) | Ops Service: | Initiate (déclencher) |
| 6 | (o) | Ops Service: | Terminate (Mettre fin) |
| 7 | (o) | Ops Service: | Conclude |
| 8 | (o) | Ops Service: | Reject |

#### 6.2.2.2.2 Attributs des services Identify, get status et status notification

**Liste des attributs de service de l'AP**

Les attributs suivants sont accessibles à l'aide des services Identify, Get Status et Status Notification. Ils ne sont pas accessible autrement.

**Manufacturer Identifier**

Le Manufacturer Identifier ('identificateur de fabricant) est composé du Vendor Name (nom de fournisseur), du Model Identifier (identificateur de modèle), de la Vendor Revision (révision du fournisseur) et, facultativement, du Serial Number (numéro de série) de l'AP. Ces attributs peuvent être lus à l'aide du service Identify.

**Vendor Name**

Cet attribut spécifie le nom du fabricant de l'AP.

**Model Identifier**

Cet attribut spécifie le modèle de l'AP.

**Vendor Revision**

Cet attribut spécifie le niveau de révision de l'AP tel que défini par le fabricant.

**Serial Number**

Cet attribut facultatif spécifie le numéro de série de l'AP.

NOTE Le numéro de série peut être utilisé comme qualificateur pour identifier un AP de façon univoque.

**AP Descriptor Reference**

Cet attribut est une référence à une description générique de l'AP. Le descripteur est un identificateur pour les caractéristiques de l'AP indépendant de son utilisation. Par conséquent, deux AP peuvent partager la même description générique. La valeur et l'ensemble admissible de valeurs pour cet attribut sont définis par l'utilisateur. Cet attribut est utilisé dans le service Initiate.

**Network Access State**

L'attribut Network Access State définit la capacité de l'AP à communiquer. Deux composants sont spécifiés dans la présente norme, à savoir Service Access Status et Operational Status. Ces attributs peuvent être lus à l'aide du service Get Status. L'AP peut choisir de les rapporter sans avoir reçu une demande, en utilisant le service Status Notification.

**Service Access Status**

Cet attribut spécifie les informations relatives à l'état des capacités de communication de l'AP.

| | |
|---|---|
| READY FOR COMMUNICATION | Tous les services peuvent être utilisés normalement. |
| LIMITED NUMBER OF SERVICES | Un nombre limité de services peut être pris en charge dans certains cas (pour de petits appareils, par exemple). |
| LOADING-NON-INTERACTING | Les définitions d'objets sont en train d'être chargées localement. Par conséquent, le service Begin Set Attributes ne peut pas être utilisé tant que la charge locale n'est pas terminée et que l'état n'a pas été changé. |
| LOADING-INTERACTING | Les définitions d'objets sont en train d'être chargées sur le réseau (à l'aide du service Set Attributes). Sur l'AR utilisée pour le chargement, il convient de n'utiliser que les services suivants: Abort, Get Attributes, Status, Set Attributes, Identify, End Set Attributes |

**OperationalStatus**

Cet attribut donne l'état opérationnel de l'AP réel, comme suit.

OPERATIONAL

PARTIALLY OPERATIONAL

NOT OPERATIONAL

NEEDS MAINTENANCE

#### 6.2.2.2.3 Attributs de gestion système

**Liste des attributs de gestion système**

Les attributs suivants sont accessibles par le biais de System Management. Ils sont utilisés par tous les ASE de l'AP pour appliquer le contrôle de flux des services confirmés (voir les diagrammes d'états de la CEI 61158-6-5). Ils ne sont pas accessibles autrement.

**List of AR Endpoint Info**

Cet attribut spécifie les identificateurs numériques et autres informations de configuration (les attributs d'AP Context) des AREP associés à l'AP.

**Numeric ID**

Cet attribut spécifie l'identificateur numérique de l'AREP.

**Configured Max FAL PDU Size Sending**

Pour les AR qui ne se segmentent pas, cet attribut spécifie le nombre maximal configuré d'octets qui peuvent être envoyés à partir de cet AREP. Sa valeur est égale à la taille maximale d'unités de données du service de liaison de données moins un qui peut être envoyée à partir de cet AREP. Pour les AR qui se segmentent, il spécifie le nombre maximal de segments de 256 octets qui peuvent être envoyés sur cet AREP.

**Configured Max FAL PDU Size Receiving**

Pour les AR qui ne se segmentent pas, cet attribut spécifie le nombre maximal configuré d'octets qui peuvent être reçus sur cet AREP. Sa valeur est égale à la taille maximale d'unités de données du service de liaison de données moins un qui peut être reçue sur cet AREP. Pour les AR qui se segmentent, il spécifie le nombre maximal de segments de 256 octets qui peuvent être reçus sur cet AREP.

**Configured Max Outstanding Services Requesting (ConfigMaxOsReq)**

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés dans le rôle de client de cet AREP. Sa valeur est configurée avant l'établissement de l'AR. Cet attribut est utilisé sur des AR client-serveur et homologue à homologue (c'est-à-dire QUB) et correspond au nombre de diagrammes d'états de transaction expéditeurs

**Configured Max Outstanding Services Responding (ConfigMaxOsRsp)**

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés pour cet AREP comme serveur. Sa valeur est configurée avant l'établissement de l'AR. Cet attribut est utilisé sur des AR client-serveur et homologue à homologue (c'est-à-dire QUB) et correspond au nombre de diagrammes d'états de transaction destinataires

**List of Supported Services**

Cet attribut spécifie les services que la FAL prend en charge comme demandeur et comme répondeur. La prise en charge comme demandeur indique que la FAL fournit à l'AP la capacité de lancer des primitives "request" confirmées et non confirmées et de recevoir des primitives de confirmation de services confirmés issues de la FAL. La prise en charge comme répondeur indique que la FAL fournit à l'AP la capacité de livrer des primitives "indication" de services confirmés à l'AP et de recevoir des primitives "response" de services confirmés issues de l'AP.

**Configured Max Data Structure Nesting Level (ConfigMaxDSNestingLevel)**

Cet attribut facultatif spécifie le nombre maximal de niveaux d'imbrication configurés pour l'AR. Il est présent seulement lorsqu'au moins deux niveaux d'imbrication sont pris en charge. C'est-à-dire que la valeur par défaut pour ce paramètre est un.

**List of In-Use AR Endpoint Info**

Cet attribut spécifie des informations dynamiques pour les AREP associés à l'AP qui participent à une AR-I établie ou qui sont impliquées dans le processus d'établissement d'AR.

**Context State**

Cet attribut spécifie l'état de l'AREP tel que vu par l'AP. Les valeurs sont:

- CLOSED
- OPEN
- OPENING-REQUESTING
- OPENING-RESPONDING

**Negotiated Max FAL PDU size sending (NegMaxPduSnd)**

Pour les AR qui ne se segmentent pas, cet attribut spécifie le nombre maximal négocié d'octets qui peuvent être envoyés à partir de cet AREP. Sa valeur est égale à la taille maximale d'unités de données du service de liaison de données moins un qui peut être envoyée à partir de cet AREP. Pour les AR qui se segmentent, il spécifie le nombre maximal négocié de segments de 256 octets qui peuvent être envoyés sur cet AREP.

**Negotiated Max FAL PDU size receiving (NegMaxPduRcv)**

Pour les AR qui ne se segmentent pas, cet attribut spécifie le nombre maximal négocié d'octets qui peuvent être reçus sur cet AREP. Sa valeur est égale à la taille maximale d'unités de données du service de liaison de données moins un qui peut être reçue sur cet AREP. Pour les AR qui se segmentent, il spécifie le nombre maximal de segments de 256 octets qui peuvent être reçus sur cet AREP.

**Negotiated Max outstanding services requesting (NegMaxOsReq)**

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés pour cet AREP comme client. Sa valeur est négociée au cours de l'établissement de l'AR afin de représenter la valeur inférieure de l'attribut Configured Max Outstanding Services Requesting local et l'attribut Configured Max Outstanding Services Responding distant.

**Negotiated Max outstanding services responding (NegMaxOsRsp)**

Cet attribut spécifie le nombre maximal de services confirmés en cours autorisés pour cet AREP comme serveur. Sa valeur est négociée au cours de l'établissement de l'AR afin de représenter la valeur inférieure de l'attribut Configured Max Outstanding Services Responding local et l'attribut Configured Max Outstanding Services Requesting distant.

**Outstanding services requesting counter (OsReqCtr)**

Cet attribut spécifie le nombre de services confirmés actuellement en cours sur l'AR comme un demandeur.

**Outstanding services responding counter (OsRspCtr)**

Cet attribut spécifie le nombre de services confirmés actuellement en cours sur l'AR comme un répondeur.

**Negotiated max data structure nesting level (NegConfigMaxDSNestingLevel)**

Cet attribut facultatif spécifie le nombre maximal de niveaux d'imbrication négociés pour l'AR. Il est présent seulement lorsqu'au moins deux niveaux d'imbrication sont pris en charge. Sa valeur est négociée comme étant la plus faible configurée pour deux AP engagés dans une communication.

**Negotiated list of supported services**

Cet attribut facultatif spécifie les services négociés que la FAL prend en charge comme un demandeur et comme un répondeur sur cette AR. La prise en charge comme demandeur indique que la FAL fournit à l'AP la capacité de lancer des primitives "request" confirmées et non confirmées et de recevoir des primitives de confirmation de services confirmés issues de la FAL. La prise en charge comme répondeur indique que la FAL fournit à l'AP la capacité de livrer des primitives "indication" de services confirmés à l'AP et de recevoir des primitives "response" de services confirmés issues de l'AP.

#### 6.2.2.2.4 Attributs de gestion d'objets

**List Of Managed AP Attributes**

Cet attribut spécifie les attributs de l'AP qui sont accessibles à l'aide des services de l'ASE Management.

**List of Supported APO Classes and Objects**

Cet attribut spécifie les définitions d'objets maintenues par l'AP.

**List Header**

Cet attribut décrit les caractéristiques des définitions d'objets maintenues par l'AP.

**Numeric ID**

cet attribut est l'identificateur numérique de l'en-tête de la liste (List Header). Sa valeur est toujours 0

**ROM / RAM Flag**

Cet attribut, lorsqu'il est TRUE, indique que des modifications apportées aux définitions d'objets sont autorisées.

**Max Name Length**

Cet attribut spécifie la longueur maximale en octets pour les noms d'objets définis pour cet AP.

**Access Protection Supported**

Cet attribut, lorsqu'il est TRUE, indique que la protection d'accès (Access Protection) est prise en charge.

**Version Of List**

Cet attribut indique la version courante de cette liste. Chaque mise à jour de la liste de définitions d'objets, initiée localement ou par le service SetAttributes, fait incrémenter le numéro de version.

**List Local Reference**

Cet attribut est une référence à la liste qui a une signification locale, telle qu'une adresse. La valeur hexadécimale FFFFFFFF indique qu'aucune référence locale n'est disponible.

**XXX Object Class Supported Constraint**

Cette contrainte sélectionne des classes d'objets spécifiées par XXX. Les attributs suivant la contrainte s'appliquent aux définitions d'objets de la classe d'objets sélectionnée.

**XXX Numeric ID**

Cet attribut est l'identificateur numérique du premier objet dans une série d'objets de la classe (ou des classes) sélectionnée par la contrainte. Tous les objets de la classe (ou des classes) sélectionnée sont contenus dans la série. Le premier Numeric ID pour les Data type est toujours 1.

**Number of XXX Entries**

Cet attribut spécifie le nombre d'objets dans la série.

**XXX Local Reference**

Cet attribut est une référence au commencement des définitions d'objets qui décrivent les objets dans la série. La référence a une signification locale, telle qu'une adresse. La valeur hexadécimale FFFFFFFF indique qu'aucune référence locale n'est disponible.

**List of Supported APO Classes and Objects**

Cet attribut spécifie les définitions d'objets maintenues par l'AP.

**List of DLSAP Addresses**

Cet attribut spécifie les adresses de DLSAP disponibles utilisables par l'AP pour localiser ses AREP.

NOTE 1 La structure des adresses de DLSAP est définie dans la CEI 61158-3-1 et dans la CEI 61158-4-1.

**List of APO Classes Supported**

Cet attribut spécifie les classes d'APO prises en charge par l'AP.

**List of Event Summary Selection Criteria**

Cet attribut facultatif spécifie un ensemble de critères de sélection booléens que l'AP est capable d'utiliser pour sélectionner des événements lors de l'accomplissement du traitement sommaire d'événements. Un ensemble de critères de sélection est spécifié dans la présente norme. Lorsqu'un AP est défini, les critères spécifiés pour cet attribut peuvent être sélectionnés dans cet ensemble ou ils peuvent être spécifiquement définis pour l'AP. La liste de critères spécifie

ENABLED Lorsqu'il est TRUE, les rapports d'événements pour les objets Event (événement) sont activés

Lorsqu'il est FALSE, les rapports d'événements pour les objets Event sont désactivés

DETECTED Lorsqu'il est TRUE, l'apparition d'un événement a été détectée

Lorsqu'il est FALSE, l'apparition d'événement n'a pas été détectée.

ACKED Lorsqu'il est TRUE, un acquittement d'événement a été reçu

Lorsqu'il est FALSE, un acquittement d'événement est en souffrance.

ACTIVE Lorsqu'il est TRUE, la détection d'événement est active pour l'objet Event

Lorsqu'il est en FALSE, la détection d'événement n'est pas active pour l'objet Event

NOTE 2 Des valeurs complémentaires peuvent être spécifiées par la présente norme dans le futur ou peuvent être définies par l'utilisateur.

#### 6.2.2.2.5 Attributs pris en charge

**List of Supported Attributes**

Cet attribut facultatif spécifie les attributs pris en charge par l'objet. Cette liste contient, au minimum, les attributs obligatoires pour la classe de l'objet.

#### 6.2.2.2.6 Services

**Subscribe**

Ce service facultatif est utilisé par les abonnés push ("push subscribers ») pour indiquer aux éditeurs push ("push publishers") leurs exigences relatives à la réception des données publiées.

**Identify**

Ce service facultatif est utilisé pour demander des informations fournisseur à l'AP.

**Get Status**

Ce service facultatif est utilisé pour demander à l'AP l'état network-visible ("visible par le réseau").

**Status Notification**

Ce service facultatif est utilisé par l'AP pour rapporter son état network-visible ("visible par le réseau").

**Initiate**

Ce service facultatif est utilisé pour ouvrir le contexte AP pour une relation entre applications.

**Terminate**

Ce service facultatif est utilisé pour fermer brutalement le contexte AP pour une relation entre applications.

**Conclude**

Ce service est utilisé pour fermer progressivement le contexte AP pour une relation entre applications.

**Reject**

Ce service est lancé en interne par l'ASE d'AP pour indiquer qu'une APDU de demande de service confirmé a été reçue avec une erreur de protocole.

### 6.2.2.3 Spécification du service de l'ASE "Application process"

#### 6.2.2.3.1 Services pris en charge

Ce paragraphe contient la définition de services qui sont propres à cet ASE. Les services définis pour cet ASE sont:

- Subscribe
- Identify
- Get Status
- Status Notification
- Initiate
- Terminate
- Conclude
- Reject

#### 6.2.2.3.2 Service Subscribe

##### 6.2.2.3.2.1 Vue d'ensemble du service

Ce service confirmé est utilisé par un abonné "push" souhaitant demander de façon dynamique à un éditeur "push" l'édition de données sélectionnées. La réponse au service contient des informations suffisantes pour que l'abonné affecte un AREP pour recevoir les données.

À la réception d'une indication du service Subscribe, un éditeur "push'" peut lancer l'édition d'un objet ou modifier les contraintes de publication relatives à un objet éditer. Les actions et les politiques engagées par un AP pour satisfaire à une indication de service Subscribe sont déterminées par l'AP lui-même et ne relèvent pas du domaine d'application de la présente norme.

Ce service est également utilisé par un abonné "push" pour indiquer à l'éditeur "push" qu'il ne s'abonne plus à l'objet édité.

NOTE 1 Il est possible ou non d' avoir d'autres abonnés qui peuvent également recevoir les données produites demandées.

NOTE 2 Les éditeurs n'ont pas nécessairement besoin de recevoir une telle demande pour éditer. Par exemple, un éditeur pourrait être préconfiguré avec la connaissance de ce qui est à éditer.

##### 6.2.2.3.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 17.

**Tableau 17 – Paramètres du service Subscribe**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Application Relationship | M | M (=) | | |
| Invoke ID | U | | | |
| Joining | S | S (=) | | |
| Published Object | M | M (=) | | |
| Schedule Interval | M | M (=) | | |
| Maximum ARs | U | U (=) | | |
| Maximum Subscriber AP Permissible Jitter | U | U (=) | | |
| Desired Publishing Start Time | U | U (=) | | |
| Earliest Publishing Start Time | U | U (=) | | |
| Latest Publishing Start Time | U | U (=) | | |
| Leaving | S | S (=) | | |
| List Of Leaving AREPs | M | M | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Data type | | | C | C (=) |
| Data Length | | | C | C (=) |
| Dedicated (c'est-à-dire: Spécialisé) | | | C | C (=) |
| List Of AR Information | | | C | C (=) |
| Publishing AREP ID | | | M | M (=) |
| DL Mapping Reference | | | M | M (=) |
| Scheduled Start Time | | | M | M (=) |

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande de service.

**Joining**

Ce paramètre est sélectionné lorsqu'un AP souhaite s'abonner à un objet édité. Le service Subscribe est envoyé par l'abonné potentiel, en utilisant l'AR sélectionnée, à l'éditeur de l'objet édité.

**Published Object**

Ce paramètre spécifie l'un des attributs-clés de l'objet édité.

**Schedule Interval**

Ce paramètre spécifie l'incrément de temps bus de terrain entre des temps successifs de début pour la séquence programmée. La granularité du temps est la même que celle définie par la couche liaison de données sous-jacente (voir CEI 61158-3-1).

**Maximum ARs**

Ce paramètre facultatif spécifie le nombre maximal d'AR sur lesquelles l'abonné souhaite recevoir les données. La valeur par défaut pour ce paramètre est un (1).

NOTE Dans certaines circonstances, un AP éditeur peut se voir demander d'éditer le même objet sur plus d'une AR. Un AP abonné peut recevoir le même objet édité d'un AP éditeur sur plus d'une AR. L'AR sur laquelle le service Subscribe est conduit peut être différente de l'AR utilisée pour éditer les données.

**Maximum Subscriber AP Permissible Jitter**

Ce paramètre facultatif spécifie la seule limite supérieure sur les deux définitions de la gigue de programmation sur laquelle l'AP abonné souhaite recevoir les données:

a) la différence de temps de début d'une séquence programmée donnée au sein de cycles successifs d'AP abonné, et

b) la différence de temps de début d'un cycle donné d'AP abonné relatif à un cycle de DLL.

La valeur de ce paramètre est toujours inférieure à la valeur du paramètre Cycle Interval. Si ce paramètre n'est pas spécifié, la gigue sera déterminée par l'éditeur. La granularité peut être la même que celle définie par la couche liaison de données (voir CEI 61158-3-1).

**Desired Publishing Start Time**

Ce paramètre facultatif spécifie le temps bus de terrain auquel il convient que l'édition commence.

**Earliest Publishing Start Time**

Ce paramètre facultatif spécifie le temps bus de terrain au plus tôt auquel l'édition peut commencer.

**Latest Publishing Start Time**

Ce paramètre facultatif spécifie le temps bus de terrain au plus tard auquel il faut que l'édition commence.

**Leaving**

Ce paramètre est à sélectionner si un abonné souhaite arrête son abonnement à un objet spécifique par un AREP donné. Le service Subscribe est envoyé à l'AP éditeur en utilisant l'AR ayant servi au rattachement.

**List Of Leaving AREPs**

Ce paramètre spécifie l'/les AREP de l'éditeur "push" que l'AP abonné veut quitter.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Data type**

Ce paramètre conditionnel identifie le type de donnée de la variable. Ce paramètre est présent si l'abonné tente de rejoindre une relation.

**Data Length**

Ce paramètre conditionnel spécifie la longueur en octets de la valeur de données devant être produite. Ce paramètre est présent si l'abonné tente d'établir une relation et si la longueur n'est pas définie pour le type de données de l'objet édité.

**Dedicated**

Ce paramètre conditionnel, lorsqu'il est TRUE, indique que la relation d'édition est spécialisée. Ce paramètre est présent si l'abonné tente d'établir une relation.

**List Of AR Information**

Ce paramètre conditionnel spécifie les informations nécessaires pour que l'abonné construise son/ses AREP de l'abonné local. Ce paramètre est présent si l'abonné tente d'établir une relation. S'il est présent, la liste n'est pas vide.

**Publishing AREP ID**

Ce paramètre spécifie l'identificateur numérique pour l'AREP utilisé pour l'édition.

**DL Mapping Reference**

Ce paramètre identifie le mapping à la DL pour l'AREP éditeur. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**Scheduled start time**

Ce paramètre spécifie l'heure effective de début de publication.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.2.3.2.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Si la demande indique qu'un abonné souhaite "se connecter", l'AP éditeur détermine localement s'il peut prendre en charge les spécificités de la demande. S'il le peut, il peut (1) reconfigurer un ou plusieurs points d'extrémité d'AR existants ou (2) allouer des ressources et créer un ou plusieurs nouveaux points d'extrémité de l'AR ou encore (3) allouer de la largeur de bande dans le programme réseau si cela s'avère nécessaire.

Si la demande indique qu'un abonné souhaite "se déconnecter", l'AP détermine localement s'il peut cesser d'éditer sur l'AR indiquée et encore prendre en charge ses autres abonnés existants. Si tel est le cas, l'AP peut libérer localement le(s) AREP et annuler le(s) programme(s) réseau correspondant(s).

Si une confirmation (+) a été retournée pour une opération "join" (Connexion), l'utilisateur de la FAL dans l'AP recevant l'APDU de réponse peut configurer ou peut créer son/ses AREP local/locaux qui serviront à recevoir la variable publiée.

Si une confirmation (+) a été retournée pour une opération "leave" (Déconnexion), l'utilisateur de la FAL dans l'AP peut libérer localement les ressources associées à l'AREP (ou aux AREP) utilisé pour l'abonnement.

Le fait que l'utilisateur de la FAL détermine si le programme spécifié par l'éditeur est suffisant relève d'une initiative locale. S'il est suffisant, il convient que l'utilisateur local émette une primitive du service Subscribe pour informer l'éditeur de son départ.

#### 6.2.2.3.3 Service Identify

##### 6.2.2.3.3.1 Vue d'ensemble du service

Les informations pour identifier un AP sont lues avec ce service.

NOTE L'attribut ProfileNumber de l'AP est émis avec le service Initiate.

##### 6.2.2.3.3.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 18.

**Tableau 18 – Identify**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Vendor Name | | | M | M (=) |
| Model Identifier | | | M | M (=) |
| Vendor Revision | | | M | M (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**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 de l'AP.

**Model Identifier**

Ce paramètre spécifie la valeur de l'attribut Model Identifier de l'AP.

**Vendor Revision**

Ce paramètre spécifie la valeur de l'attribut Vendor Revision de l'AP.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.2.3.3.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.2.3.4 Service Get status

#### 6.2.2.3.4.1 Vue d'ensemble du service

L'état de l'appareil /utilisateur est lu avec le service Get Status.

#### 6.2.2.3.4.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 19.

**Tableau 19 – Get status**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Service Access Status | | | M | M (=) |
| Operational Status | | | M | M (=) |
| Local Detail | | | U | U (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**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.

**Service Access Status**

Ce paramètre spécifie la valeur de l'attribut Service Access Status de l'AP.

**Operational Status**

Ce paramètre spécifie la valeur de l'attribut Operational Status de l'AP.

**Local Detail**

Ce paramètre spécifie l'état de l'application défini par l'utilisateur.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.2.3.4.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.2.3.5 Service Status notification

#### 6.2.2.3.5.1 Vue d'ensemble du service

Le service Status Notification est utilisé par l'AP pour rapporter spontanément son état.

#### 6.2.2.3.5.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 20.

**Tableau 20 – Status notification**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Service Access Status | M | M (=) |
| Operational Status | M | M (=) |
| LocalDetail | U | U (=) |

**Argument**

L'argument contient les paramètres de la demande de service.

**Destination DL-Address**

Ce paramètre existe seulement lorsque l'AREP correspondant le prend en charge et l'attribut Remote Address Configuration Type est FREE. Il donne l'adresse distante à laquelle il convient d'envoyer la notification de statut (Status Notification).

**Source DL-Address**

Ce paramètre existe seulement lorsque l'AREP correspondant le prend en charge et l'attribut Remote Address Configuration Type est FREE. Ce paramètre indique l'adresse distante à partir de laquelle la notification de statut (Status Notification) indiquée est envoyée.

**Service Access Status**

Ce paramètre spécifie la valeur de l'attribut Service Access Status de l'AP.

**Operational Status**

Ce paramètre spécifie la valeur de l'attribut Operational Status de l'AP.

**Local Detail**

Ce paramètre spécifie l'état de l'application défini par l'utilisateur.

#### 6.2.2.3.5.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

#### 6.2.2.3.6 Service Initiate

##### 6.2.2.3.6.1 Vue d'ensemble du service

Ce service est utilisé pour établir un contexte entre des AP engagés dans une communication pour les informations échangées sur une seule AR. Le service Initiate négocie les services de FAL pris en charge, les services maximum en cours permis, la longueur maximale des PDU et la version courante des définitions d'objets. Le niveau d'imbrication des structures de données est également négocié si les AP locaux et distants prennent en charge des niveaux d'imbrication de structures de données supérieurs à un. Les services de FAL pris en charge, les services maximum en cours permis, la longueur maximale des PDU et le niveau d'imbrication des structures de données sont des attributs de gestion système AP configurés pour l'AREP. Ce service peut être utilisé pour créer de façon dynamique l'AREP devant être utilisé si l'AREP n'existe pas.

##### 6.2.2.3.6.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 21.

**Tableau 21 – Initiate**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Version Object Definitions Calling | M | M (=) | | |
| AP Descriptor Calling | M | M (=) | | |
| Access Protection Supported Calling | M | M (=) | | |
| Password Calling | M | M (=) | | |
| Access Groups Calling | M | M (=) | | |
| Destination DL-Address | C | | | |
| DL-Mapping | C | C | | |
| Nesting Level Calling | U | U (=) | | |
| User Detail | U | U (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Version OD Called | | | M | M (=) |
| AP Descriptor Called | | | M | M (=) |
| Access Protection Supported Called | | | M | M (=) |
| Password Called | | | M | M (=) |
| Access Groups Called | | | M | M (=) |
| DL-Mapping | | | C | C |
| Nesting Level Called | | | U | U (=) |
| User Detail | | | U | U (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Error Code | | | M | M (=) |
| Max PDU Length Sending Called | | | | C |
| Max PDU Length Receiving Called | | | | C |
| FAL Features Supported Called | | | | C |

NOTE Voir la Note en 3.8.4.3.

**Argument**

Ce paramètre contient les paramètres de l'invocation de service.

**Version Object Definitions Calling**

Ce paramètre spécifie la version des définitions d'objets du client. Sa valeur est null (vide) si le client ne contient pas de définitions d'objets.

**AP Descriptor Calling**

Ce paramètre spécifie la valeur de l'attribut Descriptor Reference de l'AP appelant s'il en a un.S'il n'en a pas, sa valeur est null (vide).

**Access Protection Supported Calling**

Ce paramètre spécifie la valeur de l'attribut Access Protection Supported de l'AP appelant s'il en a un. S'il n'en a pas, sa valeur est null (vide).

**Password Calling**

Ce paramètre spécifie le mot de passe devant être utilisé pour l'accès à tous les objets du serveur sur cette AR. Si l'accès avec mot de passe n'est pas à utiliser sur cette AR, sa valeur est null

**Access Groups Calling**

Ce paramètre spécifie l'appartenance du client comme membre dans des groupes d'accès spécifiques. L'appartenance comme membre s'applique pour l'accès à tous les objets du serveur sur cette AR.

**Destination DL-Address**

Ce paramètre existe seulement lorsque l'AR correspondante le prend en charge et l'attribut Remote Address Configuration Type est FREE. Il donne l'adresse distante vers laquelle il convient d'établir la connexion.

**DL-Mapping**

Ce paramètre conditionnel spécifie les attributs de mapping de la DL pour l'AREP devant être ouvert. Il est seulement utilisé lorsque l'AREP à ouvrir n'est pas défini.

**Nesting Level Calling**

Ce paramètre facultatif indique le niveau d'imbrication pris en charge par le demandeur.

**User Detail**

Ce paramètre facultatif spécifie les informations d'utilisateur.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Version Object Definition Called**

Il convient que ce paramètre spécifie la version des définitions d'objets du serveur.

**AP Descriptor Called**

Ce paramètre spécifie la valeur de l'attribut Descriptor Reference de l'AP appelé.

**Access Protection Supported Called**

Ce paramètre spécifie la valeur de l'attribut Access Protection Supported de l'AP appelé s'il en a un. S'il n'en a pas, sa valeur est null (vide).

**Password Called**

Ce paramètre spécifie le mot de passe devant être utilisé pour l'accès à tous les objets du client sur cette AR. S'il ne faut pas que l'accès avec mot de passe soit utilisé sur cette AR, sa valeur est null

**Access Groups Called**

Ce paramètre spécifie l'appartenance du serveur comme membre dans des groupes d'accès spécifiques. L'appartenance comme membre s'applique pour l'accès à tous les objets du client sur cette AR.

**DL-Mapping**

Ce paramètre conditionnel spécifie les attributs négociés de mapping à la DL pour l'AREP devant être ouvert. Il est seulement utilisé lorsque l'AREP à ouvrir n'est pas défini.

**Nesting Level Called**

Ce paramètre facultatif indique le niveau d'imbrication pris en charge par le répondeur.

**User Detail**

Ce paramètre facultatif spécifie les informations d'utilisateur.

**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.

| Cause | Signification |
|---|---|
| Max FAL PDU Size Insufficient (Taille max des PDU de FAL insuffisante) | La longueur maximale des PDU n'est pas suffisante pour la communication. |
| Service Not Supported (Service non pris en charge) | Le service ou l'option demandé(e) n'est pas pris(e) en charge par le serveur. |
| User Initiate Denied (Lancement par l'utilisateur refusé) | L'utilisateur de la FAL refuse d'établir la connexion. |
| Version Object Definition incompatible (Version de Définition d'objet incompatible) | Les versions des définitions d'objets d'appelé et d'appelant ne sont pas compatibles. |
| Password Error (Erreur de mot de passe) | Il existe déjà une AR établie avec le même mot de passe ou bien le mot de passe n'est pas valide. |
| AP Descriptor incompatible (Descripteur d'AP incompatible) | Le descripteur n'est pas pris en charge par le serveur. |
| Other (Autre) | Cause autre que l'une de celles identifiées ci-dessus. |

**Max PDU Length Sending Called**

Il convient que ce paramètre spécifie la longueur maximale de la PDU de la FAL devant être envoyée qui peut être gérée sur cette AR.

Il convient qu'il soit émis par la FAL appelée et fasse seulement partie de la primitive Initiate.cnf primitive.

**Max PDU Length Receiving Called**

Il convient que ce paramètre spécifie la longueur maximale de la PDU de la FAL devant être reçue qui peut être gérée sur cette AR.

Il convient qu'il soit émis, s'il est pris en charge, par la FAL appelée et fasse seulement partie de la primitive Initiate.cnf.

**FAL Services Supported Called**

Il convient que ce paramètre identifie les services de l'AL facultatifs et les options prises en charge par le serveur (voir la liste d'AR).

Il convient qu'il soit émis, s'il est pris en charge, par la FAL appelée et fasse seulement partie de la primitive Initiate.cnf.

#### 6.2.2.3.6.3 Procédure du service

Ce service fonctionne par le biais d'une file d'attente.

L'utilisateur demandeur présente la primitive de demande de service à son AE de la FAL et démarre un temporisateur associé afin de surveiller la demande. L'ASE d'AP construit le corps Initiate Request APDU Body et l'achemine sur l'AR spécifiée en utilisant la primitive "request" du service AR Establish. Il crée également un diagramme d'états des transactions.

À la réception de l'Initiate Request APDU Body dans la primitive "indication" du service AR Establish, l'ASE d'AP répondeur le décode. Si une erreur de protocole intervient au cours du décodage de l'APDU Body reçu, l'ASE utilise le service AR Abort pour abandonner l'AR. S'il ne s'est pas produit d'erreur de protocole, l'ASE livre à son utilisateur une primitive "indication" du service Initiate.

Si l'utilisateur répondeur est capable de traiter la demande avec succès, l'utilisateur retourne une primitive "response (+)" du service Initiate.

Si l'utilisateur répondeur est incapable de traiter la demande avec succès, le service échoue et l'utilisateur émet une primitive "response (-)" du service Initiate indiquant la cause.

L'ASE d'AP répondeur construit un corps d'APDU de réponse de service Initiate pour une primitive " response (+)" ou un corps d'APDU d'erreur du service Initiate pour une primitive "response (-) " et l'achemine sur l'AR spécifiée en utilisant une primitive "response" du service AR Establish.

À la réception du corps d'APDU retourné contenu dans une primitive "confirmation" (+ ou -) de service AR Establish, l'ASE déclencheur livre à son utilisateur une primitive "confirmation'" du service Initiate qui spécifie le succès ou l'échec, et la cause de l'échec si échec il y a eu. Il annule également le diagramme d'états des transactions correspondant.

Cependant, si le temporisateur de transactions d'AP expire avant que l'APDU de réponse ou d'erreur correspondante ne soit reçue, l'AP peut prendre des actions correctives, qui peuvent inclure de donner l'instruction à son ASE local d'annuler le diagramme d'états des transactions. Une telle instruction donnée à l'ASE passerait par une interface définie localement.

### 6.2.2.3.7 Service Terminate

#### 6.2.2.3.7.1 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 éditeur/récepteur dans une AR.

#### 6.2.2.3.7.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 22.

**Tableau 22 – Terminate**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Locally Generated | | M |
| Terminate Identifier | M | M (=) |
| Reason Code (Code de cause) | M | M (=) |
| Terminate Detail | U | U (=) |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Locally Generated**

Il convient que ce paramètre indique si l'arrêt a été généré localement ou par le partenaire de communication.

La valeur FALSE n'est pas autorisée si le paramètre Terminate Identifier à la valeur FAL et le paramètre Reason Code a la valeur AR Error.

**Terminate Identifier**

Il convient que ce paramètre indique où la cause de l'arrêt a été détectée. Les valeurs possibles sont:

FAL USER
ASE FAL APO
ASE FAL AR
DLL

**Reason Code**

Il convient que ce paramètre spécifie la cause de l'arrêt.

Si le paramètre Terminate Identifier a la valeur USER, il convient d'appliquer les valeurs suivantes.

| | Signification |
|---|---|
| Disconnection (Déconnexion) | La connexion est libérée par l'utilisateur de la FAL. |
| Version Object Definition incompatible (Version de Définition d'objet incompatible) | Les versions d'appelé et d'appelant des définitions d'objets ne sont pas compatibles. |
| Password Error (Erreur de mot de passe) | Il existe déjà une AR établie avec le même mot de passe ou bien le mot de passe n'est pas valide. |
| Descriptor incompatible (Descripteur incompatible) | Le descripteur du serveur n'est pas pris en charge par le client. |
| Limited Services Permitted (Services limités permis) | L'AP est dans le statut logique LIMITED-SERVICES-PERMITTED. |

Si le paramètre Abort Identifier a la valeur FAL, il convient d'appliquer les valeurs suivantes.

| Cause | Signification |
|---|---|
| AR Error (Erreur d'AR) | AR en défaut |
| User Error (Erreur d'utilisateur) | Primitive de service incorrecte, inconnue ou défectueuse reçue en provenance de l'utilisateur de la FAL |
| FAL PDU Error (Erreur de PDU de FAL° | PDU de la FAL inconnue ou défectueuse reçue en provenance de l'ASE d'AR |
| Connection State Conflict AR ASE (Conflit d'état de connexion d'ASE d'AR) | Primitive de service ASE d'AR incorrecte |
| AR ASE Error (Erreur d'ASE d'AR) | Primitives de service d'ASE d'AR inconnues ou défectueuses |
| PDU Size (Taille de PDU) | La longueur de PDU dépasse la longueur maximale de PDU. |
| Feature Not Supported (Caractéristique non prise en charge) | La PDU SERVICE-REQ reçue en provenance de l'ASE d'AR et le service ou l'option ne sont pas pris en charge comme un serveur (voir l'attribut FAL Features Supported dans l'AR) |
| Unexpected service response (réponse de service inattendue) | Primitive confirmed service.rsp inattendue reçue en provenance de l'utilisateur de la FAL, ou bien CONFIRMED_SERVICE-RSP_PDU inattendue reçue en provenance de l'ASE d'AR |
| Max Services Overflow (Dépassement du maximum de services) | CONFIRMED_SERVICE-REQ_PDU reçue en provenance de l'ASE d'AR and Outstanding Services Counter Receiving ≥ Max Outstanding Services Receiving |
| Connection State Conflict (Conflit d'état de connexion) | INITIATE-REQ_PDU reçue en provenance de l'ASE d'AR |
| Service Error (Erreur de service) | Le service dans la réponse ne concorde pas avec le service dans l'indication ou le service dans la confirmation ne concorde pas avec le service dans la demande. |

Si le paramètre Terminate Identifier a la valeur AR ASE ou DLL, il convient que le paramètre Reason Code parameter soit fourni par l'ASE d'AR.

**Terminate Detail**

Ce paramètre facultatif spécifie des informations complémentaires relatives à la cause d'abandon (16 octets max.). En cas de rapports d'erreur issus de l'application, la signification est définie dans le profil.

#### 6.2.2.3.7.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.2.3.8 Service Conclude

#### 6.2.2.3.8.1 Vue d'ensemble du service

Il convient que ce service soit utilisé pour libérer progressivement une AR existante entre des AP.

#### 6.2.2.3.8.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 23.

**Tableau 23 – Conclude**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Result(+) | | | S | S (=) |
| Result(–) | | | S | S (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

Ce paramètre contient les paramètres de l'invocation du 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.

#### 6.2.2.3.8.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.2.3.9 Service Reject

#### 6.2.2.3.9.1 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 APDU a été reçue avec un code de type de service non valide. Lorsqu'un autre ASE APO détecte une erreur de protocole, il demande en interne à l'ASE d'AP de générer une Reject PDU. Ce service n'est pas utilisé pour indiquer qu'une erreur de service a été détectée par l'utilisateur.

#### 6.2.2.3.9.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 24.

**Tableau 24 – Reject**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Reject Code (Code de rejet) | M | M (=) |
| Original FAL Service Type | M | M (=) |
| Original PDU Body | U | U (=) |

**Argument**

L'argument contient les paramètres de la demande du service.

**Reject Code**

Ce paramètre indique la cause du rejet. Les valeurs de ce paramètre peuvent indiquer:

Unsupported Service
(Service non pris en charge)

AR Not Established(AR non établie)

AR Not Defined
(AR non définie)

Max Outstanding Requests Exceeded (Demandes max en cours dépassées)

Max PDU Size Exceeded (Taille max de PDU dépassée)

**Original FAL Service Type**

Ce paramètre spécifie le type de service de la demande de service rejetée.

**Original PDU Body**

Ce paramètre facultatif spécifie certaines ou toutes les données de l'APDU qui ont été rejetées. La quantité de données incluses est déterminée par l'utilisateur.

#### 6.2.2.3.9.3 Procédure du service

Le service Reject est un service qui opère à travers une file d'attente ou un buffer. Il ne peut être initié que par l'ASE AP de la FAL.

Si un ASE APO quelconque reçoit une APDU de demande confirmée qui contient une erreur de protocole, l'ASE AP construit une APDU de demande de rejet (Reject Request) et la retourne sur l'AR spécifiée.

La réception de l'APDU Reject Request, l'ASE d'AP de réception livre une primitive AR-Reject.indication par l'ASE de FAL qui a émis la primitive "request" d'envoi confirmé et il émet une confirmation négative à l'utilisateur demandeur.

### 6.2.3 ASE de relation entre applications

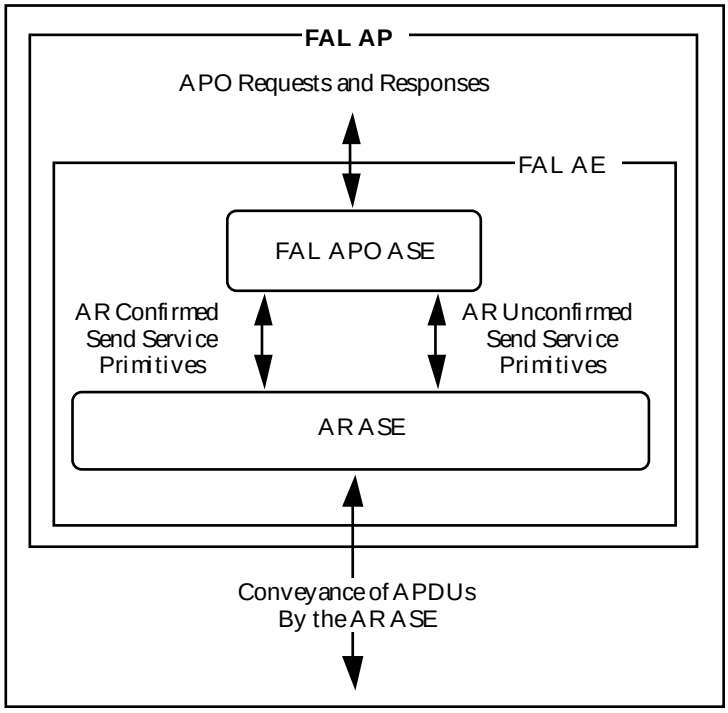
#### 6.2.3.1 Vue d'ensemble

##### 6.2.3.1.1 Généralités

Dans un système réparti, les processus application communiquent les uns avec les autres en échangeant des messages de couche application à travers des voies de communication bien définies de la couche application. Ces voies de communication sont modélisées dans la couche application (AL) de bus de terrain comme étant des relations entre applications (les AR).

Les AR sont chargées d'acheminer les messages entre des applications en fonction des caractéristiques de communication spécifiques requises par les systèmes à temps critique. Les différentes combinaisons de ces caractéristiques conduisent à la définition de différents types d'AR. Les caractéristiques des AR sont formellement définies comme étant des attributs des classes AR Endpoint (des points d'extrémité d'AR).

Les messages qui sont acheminés par les AR sont des demandes de services et des réponses de la FAL. Chacun d'eux est présenté à l'ASE d'AR pour son transfert par un élément de service application (ASE) de la FAL qui représente la classe de l'APO auquel l'accès est effectué. La Figure 1 illustre ce concept.




**Légende**

| Anglais | Français |
|---|---|
| FAL AP | AP de la FAL (Processus application de la couche application des bus de terrain) |
| APO Requests and Responses | Demandes et réponses APO |
| FAL AE | FAL AE |
| FAL APO ASE | ASE FAL APO |
| AR confirmed Send Service Primitives | Primitives de service “AR confirmed Send” |
| AR Unconfirmed Send Service Primitives | Primitives de service “ AR Unconfirmed Send” |
| AR ASE | ASE AR |
| Conveyance of APDUs By the AR ASE | L’acheminement des APDU par l’ASE AR |

**Figure 1 – L'ASE d'AR achemine des APDU entre des AP**

En fonction du type d'AR, les APDU peuvent être envoyées vers un ou plusieurs processus application destinataires reliés par l'AR. D'autres caractéristiques de l'AR déterminent la manière dont il faut transférer les APDU. Ces caractéristiques sont décrites ci-dessous.

#### 6.2.3.1.2 Contexte de point d'extrémité

Chaque AP impliquée dans une AR contient un point d'extrémité de l'AR. Chaque point d'extrémité d'AR est défini au sein de l'AE de l'AP. Sa définition, lorsqu'elle est combinée aux définitions des autres points d'extrémité, définit une AR. Afin d'assurer la comptabilité des communications parmi ou entre des points d'extrémité, chaque définition de point d'extrémité contient un jeu de caractéristiques liées à la compatibilité. Pour que l'AR fonctionne correctement, il est nécessaire que ces caractéristiques soient configurées de façon appropriée pour chaque point d'extrémité.

Les définitions de point d'extrémité contiennent également un jeu de caractéristiques qui décrivent le fonctionnement de l'AR. Ces caractéristiques, lorsqu'elles sont combinées avec celles qui sont utilisées pour spécifier la compatibilité, définissent le contexte du point d'extrémité. Le contexte du point d'extrémité est utilisé par l'ASE de l'AR pour gérer le

fonctionnement du point d'extrémité et l'acheminement des APDU. Les caractéristiques qui composent le contexte du point d'extrémité sont décrites ci-après.

#### 6.2.3.1.2.1 Rôle d'un point d'extrémité

Le rôle d'un AREP détermine le comportement admissible d'un AP en l'AREP. Un rôle d'AREP peut être celui d'un client, d'un Serveur, d'un homologue (client et/ou Serveur), un éditeur push ou pull, un abonné push ou pull, ou un gestionnaire d'édition push ou pull.

NOTE Voir le paragraphe des concepts pour la description des clients et des serveurs, et les modèles push et pull pour les éditeurs et les abonnés.

Le Tableau 25 et le Tableau 26 résument les caractéristiques et les combinaisons de chacun des rôles d'AREP.

**Tableau 25 – Acheminement de primitives de service par rôle d'AREP**

| | Client | Serveur | Homologue | Éditeur Push | Abonné Push | Gestion. Édition Pull | Éditeur Pull | Abonné Pull |
|---|---|---|---|---|---|---|---|---|
| Send Service Req | X | | X | X | | X | | |
| Recv Service Req | | X | X | | X | | X | |
| Send Service Rsp | | X | X | | | | X | |
| Recv Service Rsp | X | | X | | | X | | X |

**Tableau 26 – Combinaisons valides des rôles d'AREP impliqués dans une AR**

| Rôles d'AREP par modèle d'interaction | Client | Serveur | Homologue | Éditeur Push | Abonné Push | Gestion. Édition Pull | Éditeur Pull | Abonné Pull |
|---|---|---|---|---|---|---|---|---|
| Client/serveur | | | | | | | | |
| Client | | Oui | Oui | | | | | |
| Serveur | | | Oui | | | | | |
| Homologue | | | Oui | | | | | |
| Éditeur/Abonné | | | | | | | | |
| Éditeur Push | | | | | Oui | | | |
| Abonné Push | | | | | | | | |
| Gesti. Édition Pull | | | | | | | Oui | |
| Éditeur Pull | | | | | | | | Oui |
| Abonné "pull" | | | | | | | | |

#### 6.2.3.1.2.2 Points d'extrémité AR dédiés

Les points d'extrémité AR qui sont dédiés fournissent leurs services directement à l'utilisateur de FAL. Même si leur comportement est le même que celui des points d'extrémité non dédiés, les APDU qu'ils acheminent ne contiennent pas d'informations de contrôle de protocole spécifique à un service autres que le champ de contrôle de l'AR.

Les utilisateurs de FAL configurés pour des AR dédiées construisent et transfèrent leurs données sans utiliser les services des autres ASE de FAL. Le format et le contenu des données d'utilisateur acheminées sur des AR dédiées sont connus seulement par l'intermédiaire de la configuration.

#### 6.2.3.1.2.3 Cardinalité

Du point de vue d'un point d'extrémité client ou éditeur, la cardinalité d'une AR spécifie le nombre des processus applications distants qui sont impliqués dans une AR. La cardinalité n'est jamais exprimée à partir du point de vue d'un serveur ou d'un abonné.

Lorsque la cardinalité est exprimée du point de vue d'un point d'extrémité client ou homologue, les AR sont toujours des relations un à un (1:1). Les clients ne sont jamais capables d'émettre une demande et d'attendre des réponses provenant de plusieurs serveurs.

Lorsque "la cardinalité" est exprimée du point de vue d'un point d'extrémité éditeur, les AR indiquent que plusieurs abonnés sont pris en charge. Ces AR sont des relations un à plusieurs. Les AR qui sont un à plusieurs permettent la communication entre une application et un groupe de (un ou plusieurs) applications. Elles sont souvent appelées "multipartites" et "de multidiffusion". Pour prendre en charge le modèle “pull” d'édition, ces AR fournissent une voie de gestion entre le gestionnaire d'édition et l'éditeur.

Dans les AR du type 1 à plusieurs, le point d'extrémité éditeur identifie les point d'extrémités distants en utilisant un identificateur de groupe, plutôt que d'identifier chacun d’entre eux individuellement comme c'est le cas avec les AR du type un à un. Cela permet aux abonnés de rejoindre et de quitter des AR sans perturber le contexte du point d'extrémité éditeur, car leurs identités individuelles ne sont pas connues du point d'extrémité éditeur. Cependant, l'application éditeur peut se rendre compte de leur participation, mais cette information ne fait pas partie du contexte du point d'extrémité de l’AR.

#### 6.2.3.1.2.4 Modèle d'acheminement

##### 6.2.3.1.2.4.1 Généralités

Le modèle d'acheminement définit la manière dont les APDU sont envoyées entre les points d'extrémité d'une AR. Trois caractéristiques sont utilisées pour définir ces transferts:

trajets d'acheminement

politique de déclenchement, et

politique d'acheminement.

##### 6.2.3.1.2.4.2 Trajets d'acheminement

Le but des ASE d'AR est de transférer des informations entre des points d'extrémité de l’AR. Ce transfert d'informations a lieu sur les trajets d'acheminement d'une AR. Un trajet d'acheminement représente un trajet de communication unidirectionnel qui est utilisé par un point d'extrémité pour une entrée ou une sortie.

Afin de prendre en charge le rôle du processus application, le point d'extrémité est configuré avec un seul ou avec deux trajets d’acheminement. Les points d'extrémité qui ne font que seulement envoyer ou seulement recevoir sont configurés pour un trajet d'acheminement soit d'envoi, soit de réception, alors que ceux qui font les deux sont configurés pour les deux types de trajets. Les AR avec un seul trajet d'acheminement sont dites "unidirectionnelles", alors que celles avec deux trajets d'acheminement sont dites "bidirectionnelles".

Les AR unidirectionnelles sont capables d'acheminer seulement des demandes de services. Pour acheminer des réponses de services, une AR bidirectionnelle est nécessaire. Par conséquent, les AR unidirectionnelles prennent en charge le transfert de services non confirmés dans un sens seulement, alors que les AR bidirectionnelles prennent en charge le transfert des services non confirmés et confirmés déclenché par un seul point d'extrémité ou par les deux points d'extrémité.

##### 6.2.3.1.2.4.3 Politique de déclenchement

La politique de déclenchement indique le moment où les APDU sont émises par la couche liaison de données sur le réseau.

Le premier type est appelé "user-triggered" (c'est-à-dire: "déclenché par un utilisateur"). Les AREP déclenchés par l'utilisateur présentent les APDU de la FAL à la couche liaison de données en vue de leur émission à la première occasion.

Le second type est appelé "network-scheduled" (c'est-à-dire: "programmé réseau"). Les AREP programmés réseau présentent les APDU de la FAL à la couche liaison de données en vue de leur émission conformément à un programme configuré par le gestionnaire.

Ce mécanisme de planification réseau peut être cyclique ou unitaire.

Les AR programmées réseau peuvent aussi acheminer avec le service Compil des demandes et des réponses de manière asynchrone si la couche sous-jacente a une largeur de bande disponible. Ces transferts ont lieu en plus des transferts programmés et, ce, sans être déclenchés à partir du programme réseau.

#### 6.2.3.1.2.4.4 Politique d'acheminement

La politique d'acheminement indique si les APDU sont transférées selon un modèle de buffer ou un modèle de file d'attente. Ces modèles décrivent la méthode d'acheminement des APDU de l'expéditeur vers le destinataire.

Les AR buffer contiennent les trajets d'acheminement qui ont un seul buffer dans chaque point d'extrémité. Les mises à jour du buffer source sont acheminées vers le buffer de destination conformément à la politique de déclenchement de l'AR. Les mises à jour pour l'un ou l'autre des buffers remplacent le contenu par les nouvelles données. Dans les AR buffer, les données non acheminées ou non livrées qui avaient été remplacées sont perdues. De plus, les données contenues dans un buffer peuvent être lues plusieurs fois sans destruction du contenu.

Les AR file d'attente contiennent les trajets d'acheminement qui sont représentés comme une file d'attente entre les points d'extrémité. Les AR file d'attente acheminent les données en utilisant une file d'attente FIFO (First In, First Out, c'est-à-dire: premier entré, premier sorti). Les AR file d'attente ne sont pas écrasées; les nouvelles entrées sont placées en file d'attente jusqu'à ce qu'elles puissent être acheminées et livrées.

Si une file d'attente est pleine, un nouveau message ne sera pas placé dans la file d'attente.

NOTE Les services d'acheminement d'AR sont décrits de manière abstraite afin qu'ils puissent être mis en œuvre et opérer en utilisant des buffers ou des files d'attente. Ces services peuvent être mis en œuvre de plusieurs façons. Par exemple, ils peuvent être mis en œuvre d'une manière fournissant la capacité de charger le buffer/la file d'attente et ensuite de l'afficher en vue de son transfert par la couche liaison de données sous-jacente. Ou bien, ces services peuvent être mis en œuvre d'une manière telle que ces capacités soient combinées afin de pouvoir charger et transférer le buffer/la file d'attente dans une seule et même demande. Du côté réception, ces services peuvent être mis en œuvre en livrant les données lorsqu'elles sont reçues ou en indiquant leur réception et en permettant à l'utilisateur de les récupérer dans une opération distincte. Une autre option est de demander à l'utilisateur de détecter que la mise à jour du buffer ou de la file d'attente a eu lieu.

### 6.2.3.1.3 Services de communications sous-jacents

#### 6.2.3.1.3.1 Généralités

L'ASE d'AR achemine les APDU de la FAL en utilisant les capacités de la couche liaison de données sous-jacente. Plusieurs caractéristiques sont utilisées pour décrire ces capacités. Le présent paragraphe donne une description de chacune de celles-ci. Ces caractéristiques sont spécifiques au mapping de la liaison de données défini dans la CEI 61158-6-5. Leur spécification précise peut y être consultée.

#### 6.2.3.1.3.2 Services sans connexion et orientés connexion

La couche sous-jacente peut prendre en charge les points d'extrémité de l'AR en fournissant des services sans connexion ou des services orientés connexion. Les points d'extrémité configurés pour opérer avec des services orientés connexion peuvent être capables d'ouvrir et de fermer des connexions si les connexions qu'elles utilisent ne sont pas préconfigurées pour être ouvertes.

NOTE Sans connexion et orienté connexion sont des termes de couche de liaison. Se référer à la CEI 61158-3-1 et à la CEI 61158-4-1 pour la définition.

#### 6.2.3.1.3.3 Services en buffer et en file d'attente

La couche sous-jacente peut prendre en charge les points d'extrémité de l'AR en fournissant des services en buffer ou en file d'attente. Ces services peuvent être utilisés pour mettre en œuvre des buffers ou des files d'attente exigé(e)s par certaines classes de points d'extrémité.

#### 6.2.3.1.3.4 Transferts cycliques et acycliques

La couche sous-jacente peut prendre en charge les points d'extrémité de l'AR en fournissant des services cycliques ou acycliques.

La couche sous-jacente peut offrir ses services pour permettre le transfert acyclique des APDU simultanément au transfert des APDU entre les deux points d'extrémité.

#### 6.2.3.1.3.5 Qualité de service

La couche sous-jacente peut fournir plus d'une qualité de service. Lorsque cela se produit, il convient que l'AREP soit configuré pour utiliser la qualité de service appropriée pour les APDU qu'il achemine. Les paramètres de qualité de service pour les AREP sont contenus dans le mapping de la couche liaison de données dans la CEI 61158-6-5.

#### 6.2.3.1.3.6 Segmentation

La FAL a la capacité de segmenter les données utilisateur qu'elle achemine. En outre, lorsque la couche sous-jacente fournit la segmentation, la longueur maximale des données acheminées par la FAL dans une seule APDU augmentera en conséquence.

#### 6.2.3.1.3.7 Format d'adresse

La couche sous-jacente peut prendre en charge plus d'un format d'adresse. Lorsqu'elle le fait, il convient que tous les points d'extrémité d'une AR soient mappés avec des formats d'adresse compatibles. Les mappings de DL pour les AREP sont spécifiés dans la CEI 61158-6-5 pour la couche liaison de données (CEI 61158-3-1).

#### 6.2.3.1.3.8 Cohérence temporelle (Timeliness)

Des points d'extrémité de l'AR peuvent être définis pour prendre en charge les informations de cohérence temporelle relatives à leurs transferts telles que calculées par la couche liaison de données. Les informations de cohérence temporelle sont reçues de la couche liaison de données et transmises à l'utilisateur de la FAL comme un paramètre du service de FAL associé. Les points d'extrémité de l'AR éditeur et abonné prennent en charge la cohérence temporelle transparente ("transparent"), résidence ("residence"), synchronisée ("synchronized") et mise à jour ("update"). Ces types de cohérence temporelle sont décrits dans les définitions de chacun de ces types de points d'extrémité ci-dessous. Le mécanisme utilisé par la couche liaison de données pour calculer la cohérence temporelle de données produites est défini dans la CEI 61158-3-1 et dans la CEI 61158-4-1. Les attributs définis par le Modèle de l'AR pour prendre en charge la cohérence temporelle sont définis dans les attributs de Mapping de la DLL pour les AREP éditeurs/abonnés dans la CEI 61158-6-5.

#### 6.2.3.1.3.9 Détection de doublons de PDU de la FAL

Les points d'extrémité de l'AR peuvent être définis pour prendre en charge la détection de la réception d'APDU des FAL dupliquées. En variante, une indication peut être reçue de la couche liaison de données pour prendre en charge cette capacité et transmise à l'utilisateur de la FAL comme un paramètre du service de FAL associé. Les attributs définis par le Modèle d'AR pour prendre en charge cette capacité sont définis dans les attributs de mapping de la DLL dans la CEI 61158-6-5.

#### 6.2.3.1.4 Établissement d'une AR

Pour qu'un point d'extrémité de l'AR soit utilisé par un processus application, il faut que l'AR correspondante soit active. Lorsqu'une AR est activée, elle est dite "établie".

L'établissement d'AR peut se produire de l'une de trois façons. Premièrement, les AR peuvent être préétablis. "Préétablie" signifie que l'AE qui maintient le contexte du point d'extrémité est créée lorsque l'AP est connectée au réseau. Dans ce cas, les communications parmi les applications impliquées dans l'AR peuvent avoir lieu sans avoir à établir d'abord l'AR de manière explicite. Toute AR peut être définie pour être préétablie.

Deuxièmes, les AR peuvent être prédéfinies, mais pas préétablies. 'Prédéfinie" signifie que les caractéristiques de l'AR sont connues de chaque point d'extrémité, mais que leurs contextes n'ont pas été synchronisés pour le fonctionnement. Dans ce cas, l'utilisation du service Establish de la FAL est requise pour les synchroniser et les ouvrir pour le transfert de données.

Dans la troisième, les AR peuvent requérir une définition et un établissement dynamiques. Dans ce cas, les définitions sont à créer pour chacun des AREP en utilisant le service Create. Après la création, elles sont établies en utilisant les services Establish si elles n'avaient pas été créées comme étant "établies".

#### 6.2.3.1.5 Classes des relations entre applications

Les AREP sont définis avec une combinaison de caractéristiques pour former différentes classes de relations AR.

### 6.2.3.2 Spécifications pour la classe de points d'extrémité des relations entre applications

#### 6.2.3.2.1 Modèle formel

Le modèle formel du point d'extrémité de l'AR définit les caractéristiques communes à tous les points d'extrémité de l'AR. Cette classe ne peut pas être instanciée. Elle n'est présente que pour l'héritage de ses attributs et de ses services par ses sous-classes, chacune de celles-ci étant spécifiée dans un paragraphe de la présente norme.

Tous les attributs des points d'extrémité de l'AR sont accessibles par la gestion système (voir la future CEI 61158-7), et par les services de l'ASE Object Management (Gestion d'objets) définis dans la présente norme.

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | | AR ENDPOINT | |
| **CLASS ID:** | | | 32 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTS DE LA GESTION SYSTÈME:** | | | |
| 1 | (m) | Attribut: | Local AP |
| 2 | (m) | Attribut: | FAL Revision |
| 3 | (m) | Attribut: | Dedicated (TRUE, FALSE) |
| 4 | (o) | Attribut: | Transfer Syntax |
| 5 | (o) | Attribut: | List Of Supported Attributes |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | AR-Unconfirmed Send |
| 2 | (o) | OpsService: | AR-Confirmed Send (Envoi de services confirmés d'AR) |
| 3 | (o) | OpsService: | AR-Establish |
| 4 | (o) | OpsService: | AR-DeEstablish |
| 5 | (o) | OpsService: | AR-Abort |

| | | | |
|---|---|---|---|
| 6 | (o) | OpsService: | AR-Compel |
| 7 | (o) | OpsService: | AR-Get Buffered Message |
| 8 | (o) | OpsService: | AR-Schedule Communication |
| 9 | (o) | OpsService: | AR-Cancel Scheduled Sequence |
| 10 | (o) | OpsService: | AR-Get DL-Time |
| 11 | (o) | OpsService: | AR-Status |
| 12 | (o) | OpsService: | AR-XON-OFF |
| 13 | (o) | OpsService: | AR-RemoteRead |
| 14 | (o) | OpsService: | AR-RemoteWrite |

#### 6.2.3.2.2 Attributs de la gestion système

**Local AP**

Cet attribut identifie l'AP attaché ou configuré pour utiliser l'AREP en utilisant une référence locale.

**FAL Revision**

Cet attribut spécifie le niveau de révision du protocole de la FAL utilisé par cette point d'extrémité. Le niveau de révision est consigné dans l'en-tête de l'AR de toutes les PDU des FAL émises

**Dedicated**

L'attribut spécifie, lorsqu'il est TRUE, que le point d'extrémité est dédié. Lorsqu'il est mis à TRUE, les services de l'ASE d'AR sont accessibles directement à l'utilisateur de la FAL

**Transfer Syntax**

Cet attribut facultatif identifie les règles de codage devant être utilisées sur l'AR. Lorsqu'il est absent, c'est la valeur par défaut de Transfert Syntax (syntaxe de transfert) de FAL selon la présente norme qui est utilisée.

**List of Supported Attributes**

Cet attribut facultatif spécifie les attributs pris en charge par l'objet. Cette liste contient, au minimum, les attributs obligatoires pour la classe de l'objet.

#### 6.2.3.2.3 Services

Tous les services définis pour cette classe sont facultatifs. Lorsqu'une instance de la classe est définie, l'un au moins de ces services est à sélectionner.

**AR-Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer un service non confirmé.

**AR-Confirmed Send**

Ce service facultatif est utilisé pour envoyer un service confirmé.

**AR-Establish**

Ce service facultatif est utilisé pour établir (ouvrir) une AR.

**AR-DeEstablish**

Ce service facultatif est utilisé par les utilisateurs clients ou homologues pour demander l'arrêt progressif d'une AR de type un à un.

**AR-Abort**

Ce service facultatif est utilisé pour abandonner (mettre brutalement fin à) une AR.

**AR-Compel**

Ce service facultatif est utilisé par l'utilisateur de la FAL pour demander à l'ASE de l'AR d'acheminer un message qui avait été différé jusqu'à ce qu'il soit libéré de façon explicite.

**AR-Get Buffered Message**

Ce service facultatif est utilisé pour demander à l'AR de récupérer un message qui est actuellement maintenu dans un buffer de la couche liaison de données locale.

**AR-Schedule Communication**

Ce service facultatif est utilisé pour programmer une séquence de DL-COMPEL pour une relation particulière.

**AR-Cancel Scheduled Sequence**

Ce service facultatif est utilisé pour annuler une séquence existante qui avait été précédemment programmée.

**AR-Get DL-Time**

Ce service facultatif est utilisé pour accéder au service DL-Time de la couche liaison de données.

**AR-Status**

Ce service facultatif est utilisé pour indiquer à l'utilisateur d'AR qu'un événement interne de l'AREP, tel que l'émission d'un buffer déclenché par un réseau, s'est produit.

**AR-XON-OFF**

Ce service facultatif entraîne la suspension ou la reprise du flux de données sur une AR spécifiée.

**AR-RemoteRead**

Ce service facultatif est utilisé pour demander à l'AR de récupérer un message qui est actuellement maintenu dans un buffer de la couche liaison de données distante.

**AR-RemoteWrite**

Ce service facultatif est utilisé pour déclencher un échange de données, après avoir écrit une valeur dans un buffer local.

#### 6.2.3.3 Spécifications des services d'ASE de relation entre applications

##### 6.2.3.3.1 Services pris en charge

Ce paragraphe contient la définition des services qui sont propres à cet ASE. Les services définis pour cet ASE sont:

AR-Unconfirmed Send

AR-Confirmed Send

AR-Establish

AR-DeEstablish

AR-Abort

AR-Compel

AR-Get Buffered Message

AR-Schedule Communication

AR-Cancel Scheduled Sequence

AR-Status

AR-XON-OFF

AR-RemoteRead

AR-RemoteWrite

Les services AR-Confirmed Send, AR-Establish et AR-DeEstablish contiennent le corps du PDU de FAL comme partie intégrante du paramètre Result dans les primitives "response" et "confirmation". Le corps de PDU de FAL peut contenir les réponses soit positive, soit négative, retournées par l'utilisateur de FAL de manière transparente à l'ASE de l'AR. Par conséquent, ces services ont un seul paramètre "Result" au lieu des paramètres distincts Result(+) et Result(-) qui sont communément utilisés pour acheminer les réponses positives et négatives retournées par l'utilisateur de FAL.

#### 6.2.3.3.2 Service "AR-unconfirmed send"

##### 6.2.3.3.2.1 Vue d'ensemble du service

Ce service est utilisé pour envoyer des APDU de demande de service AR-Unconfirmed pour les ASE d'APO de la FAL. Le service "AR-Unconfirmed Send" peut être demandé par un des deux points d'extrémité d'une AR bidirectionnelle de type un à un, au niveau du point d'extrémité serveur d'une AR unidirectionnelle de type un vers un ou au niveau du point d'extrémité éditeur d'une AR de type un vers plusieurs.

NOTE Ce service est décrit de manière abstraite afin qu'il soit capable d'opérer avec des AR qui acheminent des APDU de FAL par des buffers ou par des files d'attente. Ce service peut être mis en œuvre d'une manière fournissant la capacité de charger le buffer/la file d'attente et ensuite de l'afficher en vue de son transfert par la couche liaison de données sous-jacente. En variante, ce service peut être mis en œuvre d'une manière telle que ces capacités soient combinées afin de permettre à l'utilisateur de charger le buffer/la file d'attente et de demander son transfert en une seule et même opération.

##### 6.2.3.3.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 27.

**Tableau 27 – AR-Unconfirmed send**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Remote DLSAP Address | C | C |
| FAL Service Type | M | M (=) |
| FAL APDU Body | M | M (=) |
| Timeliness | | C |
| Duplicate FAL PDU Body | | C |

**Argument**

L'argument contient les paramètres de la demande de service.

**Remote DLSAP Address**

Ce paramètre conditionnel spécifie l'adresse de DLSAP de destination dans la demande et l'adresse de DLSAP source dans l'indication. Il est présent si l'AREP est configuré avec un attribut Remote Address Configuration Type mis à FREE.

**FAL Service Type**

Ce paramètre spécifie le type du service acheminé.

**FAL APDU Body**

Ce paramètre spécifie le corps dépendant du service pour l'APDU

**Timeliness**

Ce paramètre conditionnel indique le statut de cohérence temporelle tant local que distant du corps d'APDU de FAL contenu dans la primitive "request" du service AR-Unconfirmed Send. Ce paramètre est présent si Timeliness est pris en charge dans les attributs de Mapping à la DL de l'AREP. Les valeurs associées à ce paramètre sont définies dans la description des attributs de mapping à la DLL dans la CEI 61158-6-5.

**Duplicate FAL PDU Body**

Ce paramètre conditionnel indique si, oui ou non, la réception d'une PDU de FAL dupliquée a été détectée par la couche liaison de données. Il est présent s'il est pris en charge dans les attributs de mapping à la DL de l'AREP.

#### 6.2.3.3.2.3 Procédure du service

Le service "AR-Unconfirmed Send" est un service qui opère à travers une file d'attente ou un buffer.

L'ASE de la FAL demandeur présente à son ASE d'AR une primitive "request" de service AR-unconfirmed send. L'ASE d'AR construit une APDU de demande de service AR-Unconfirmed Send (envoi de services non confirmés d'AR). Si l'AREP local pour l'AR spécifiée prend en charge la cohérence temporelle, l'ASE d'AR indique la cohérence temporelle d'édition et d'émission dans les APDU.

Si l'AREP est placé en file d'attente, l'ASE d'AR met en file d'attente l'APDU en vue de sa présentation à la couche inférieure. Si l'AR est en buffer, l'ASE d'AR remplace le précédent contenu du buffer par l'APDU contenue dans la primitive de service.

Si l'AREP est déclenché par l'utilisateur, l'ASE d'AR demande immédiatement à la couche inférieure de transférer l'APDU. Si l'AR est programmée par un réseau, l'ASE d'AR demande à la DL de transférer les données à l'heure programmée. Le mapping à la liaison de données indique la manière dont l'ASE dd l'AR coordonne ses demandes pour émettre les données avec la couche liaison de données.

NOTE Le programme d'émission est géré par la couche sous-jacente, pas par l'ASE de l'AR. Se référer à la CEI 61158-3-1 et à la CEI 61158-4-1 pour de plus amples détails.

À la réception de l'APDU de demande de service AR-Unconfirmed Send, l'ASE d'AR destinataire livre une primitive AR-Unconfirmed send indication à l'ASE de FAL approprié comme indiqué dans le paramètre FAL Service Type. Si l'AREP destinataire prend en charge la cohérence temporelle, l'ASE d'AR inclut les paramètres de cohérence temporelle reçus en provenance de la couche liaison de données dans la primitive "indication".

### 6.2.3.3.3 Service AR-confirmed send

#### 6.2.3.3.3.1 Vue d'ensemble du service

Ce service est utilisé pour envoyer des APDU de demande et de réponse confirmées pour les ASE d'APO de la FAL. Le service "AR-Confirmed Send" peut être demandé au niveau de points d'extrémité clients et homologues des AR bidirectionnelles de type un vers un.

NOTE Ce service est décrit de manière abstraite afin qu'il soit capable d'opérer avec des AR qui acheminent des APDU de FAL par des buffers ou par des files d'attente. Ce service peut être mis en œuvre d'une manière fournissant la capacité de charger le buffer/la file d'attente et ensuite de l'afficher en vue de son transfert par la couche liaison de données sous-jacente. En variante, ce service peut être mis en œuvre d'une manière telle que ces capacités soient combinées afin de permettre à l'utilisateur de charger le buffer/la file d'attente et de demander son transfert en une seule et même opération.

#### 6.2.3.3.3.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 28.

**Tableau 28 – AR-Confirmed send**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Destination DL-Address | C | C | | |
| FAL Service Type | M | M (=) | | |
| FAL APDU Body | M | M (=) | | |
| Result (Résultat) | | | M | M (=) |
| Invoke ID | | | | U (=) |
| Source DL-Address | | | C | C |
| FAL Service Type | | | M | M (=) |
| FAL APDU Body | | | M | M (=) |
| Timeliness | | | | C |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Destination DL-Address**

Ce paramètre conditionnel spécifie l'adresse DL de destination. Il n'est présent que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE.

NOTE 1 Tous les services confirmés ne prennent pas en charge ce paramètre. L'utilisation de ce paramètre lors de l'utilisation de services confirmés relève d'une initiative locale.

**FAL Service Type**

Ce paramètre spécifie le type du service acheminé.

**FAL APDU Body**

Ce paramètre spécifie le corps dépendant du service pour l'APDU

**Result**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source DL-Address**

Ce paramètre conditionnel spécifie l'adresse DL de source. Il n'est présent que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE.

NOTE 2 Tous les services confirmés ne prennent pas en charge ce paramètre. L'utilisation de ce paramètre avec des services confirmés relève d'une initiative locale.

**FAL Service Type**

Ce paramètre spécifie le type du service acheminé.

**FAL APDU Body**

Ce paramètre spécifie le corps dépendant du service pour l'APDU

**Timeliness**

Ce paramètre conditionnel indique la cohérence temporelle de la mise à jour dans la FAL. Ce paramètre est présent si Timeliness est pris en charge dans les attributs de Mapping à la DL

de l'AREP. Les valeurs associées à ce paramètre sont définies dans la description des attributs de mapping à la DLL dans la CEI 61158-6-5.

#### 6.2.3.3.3.3 Procédure du service

Le service "AR-Confirmed Send" est un service qui opère à travers une file d'attente ou un buffer.

L'ASE de FAL demandeur présente à son ASE d'AR une primitive "request" de service AR-Confirmed Send. L'ASE d'AR ASE crée un diagramme d'états de transactions pour commander l'invocation du service.

L'ASE d'AR construit une APDU de demande de service AR-Confirmed Send. Si l'AREP local pour l'AR spécifiée prend en charge la cohérence temporelle, l'ASE d'AR indique la cohérence temporelle d'édition et d'émission dans les APDU.

Si l'AREP est placé en file d'attente, l'ASE d'AR met en file d'attente l'APDU en vue de sa présentation à la couche inférieure. Si l'AR est en mode buffer, l'ASE d'AR remplace le précédent contenu du buffer par l'APDU contenue dans la primitive de service.

Si l'AREP est déclenché par l'utilisateur, l'ASE d'AR demande immédiatement à la couche inférieure de transférer l'APDU. Si l'AR est programmée par un réseau, l'ASE d'AR demande à la DL de transférer les données à l'heure programmée. Le mapping à la liaison de données indique la manière dont l'ASE d'AR coordonne ses demandes pour émettre les données avec la couche liaison de données.

NOTE Le programme d'émission est géré par la couche sous-jacente, pas par l'ASE d'AR. Se référer à la CEI 61158-3-1 et à la CEI 61158-4-1 pour de plus amples détails.

À la réception de l'APDU de demande de service AR-Confirmed Send, l'ASE d'AR destinataire délivre une primitive "indication" de service AR-Confirmed Send à l'ASE de FAL approprié comme indiqué dans le paramètre FAL Service Type. Si l'AREP destinataire prend en charge la cohérence temporelle, l'ASE d'AR inclut les paramètres de cohérence temporelle reçus en provenance de la couche liaison de données dans la primitive "indication".

L'ASE de FAL répondeur présente à son ASE d'AR une primitive de réponse d'envoi de services confirmés. L'ASE d'AR construit une APDU de réponse d'envoi de services confirmés.

Si l'AREP est placé en file d'attente, l'ASE d'AR met en file d'attente l'APDU en vue de sa présentation à la couche inférieure. Si l'AR est en buffer, l'ASE d'AR remplace le précédent contenu du buffer par l'APDU contenue dans la primitive de service.

Si l'AREP est déclenché par l'utilisateur, l'ASE d'AR demande immédiatement à la couche inférieure de transférer l'APDU. Si l'AR est programmée par un réseau, l'ASE d'AR demande à la DL de transférer les données à l'heure programmée. Le mapping à la liaison de données indique la manière dont l'ASE d'AR coordonne ses demandes pour émettre les données avec la couche liaison de données.

À la réception de l'APDU de réponse d'envoi d services confirmés, l'ASE d'AR destinataire utilise les informations d'identification contenues dans l'APDU de réponse pour associer la réponse avec la demande appropriée et annuler le diagramme d'états de transaction associé. L'ASE d'AR livre à l'ASE de FAL demandeur une primitive "confirmation" de service AR-Confirmed Send. Si l'AREP destinataire prend en charge la cohérence temporelle, l'ASE d'AR inclut les paramètres de cohérence temporelle reçus en provenance de la couche liaison de données dans la primitive "confirmation".

Si le temporisateur expire avant que l'ASE d'AR expéditeur ne reçoive l'APDU de réponse, l'ASE d'AR annule le diagramme d'états de transaction associé et livre à l'ASE de FAL demandeur une primitive "confirmation(-)" de service AR-Confirmed Send.

#### 6.2.3.3.4 Service AR-establish

##### 6.2.3.3.4.1 Vue d'ensemble du service

Ce service confirmé opère par paires entre deux points d'extrémité d'AR pour synchroniser leurs contextes et les activer pour le transfert des APDU.

Le contexte de point d'extrémité peut être créé pendant l'établissement ou avant l'établissement par l'intermédiaire de la gestion système. Une fois qu'il est défini, les services Set Attributes et Get Attributes peuvent être utilisés pour mettre à jour et coordonner encore plus les contextes des points d'extrémité.

##### 6.2.3.3.4.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 29.

**Tableau 29 – Service AR-Establish**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Remote DL-address | C | C (=) | | |
| User Data | U | U (=) | | |
| Result | | | M | M (=) |
| User Data | | | U | U (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**AREP**

Ce paramètre spécifie des informations suffisantes pour identifier localement l'AREP devant être établi.

**Remote DL-address**

Ce paramètre conditionnel identifie l'adresse DL distante. Si l'AR devant être établie est prise en charge par une connexion DL, l'adresse DL est une adresse de DLCEP. Sinon, l'adresse DL est une adresse de DLSAP. Il est présent lorsque l'attribut Remote Address Configuration Type de l'AREP est FREE.

**User Data**

Ce paramètre facultatif spécifie les données fournies par un utilisateur qui sont à acheminer avec la demande de service.

**Result**

Ce paramètre indique que la demande de service a réussi ou échoué.

**AREP**

Ce paramètre spécifie des informations suffisantes pour identifier localement l'AREP devant être établi.

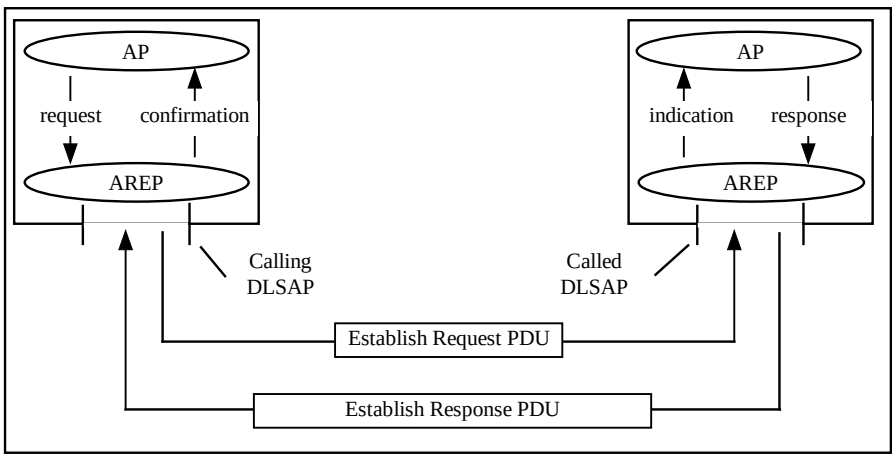
**User Data**

Ce paramètre facultatif spécifie les données fournies par un utilisateur qui sont à acheminer avec la réponse de service.

#### 6.2.3.3.4.3 Procédure du service

##### 6.2.3.3.4.3.1 Établissement d'AR de type un vers un

Lorsqu'il est utilisé à l'appui des AREP de type un à un, le service AR-Establish fait que des PDU Establish sont échangées entre les DLSAP d'appelant et d'appelé (voir Figure 2).




The Establish Request PDU is sent from the calling DLSAP AP to a called DLSAP.

The APDU contains identifiers for the AREPs to be related.

If the called AP processes the establishment request, its AREP returns an Establish Response PDU.

**Légende**

| Anglais | Français |
|---|---|
| The Establish Request PDU is sent from the calling DLSAP AP to a called DLSAP.<br>The APDU contains identifiers for the AREPs to be related.<br>If the called AP processes the establishment request, its AREP returns an Establish Response PDU. | La PDU de demande de service Establish est envoyée de l'AP de DLSAP appelant vers un DLSAP appelé.<br>Les APDU contiennent des identificateurs que les AREP devant être reliés.<br>Si l'AP appelé traite la demande d'établissement, son AREP retourne une PDU de réponse Establish. |
| request | demande |
| response | réponse |
| Calling DLSAP | DLSAP appelant |
| Called DLSAP | DLSAP appelé |
| Establish Response PDU | PDU de réponse de l'établissement |
| AREP | AREP |
| AP | AP |
| Establish Request PDU | PDU de demande de l'établissement |

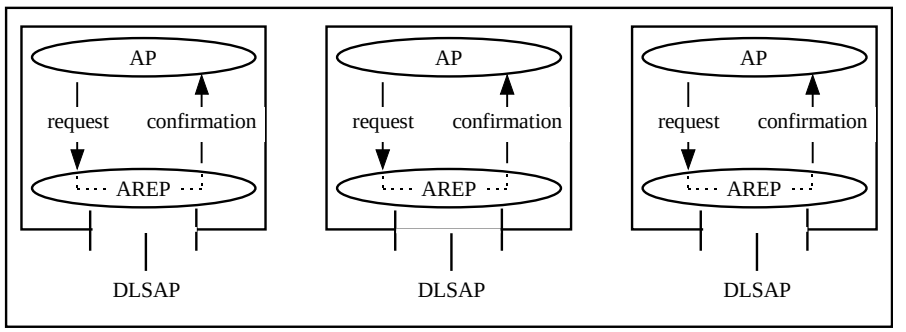
**Figure 2 – Établissement d'AR de type un vers un**

À la réception d'une primitive "request" de service 'AR-Establish, la FAL appelante émet une APDU de demande Establish vers l'ASE d'AP appelé qui gère l'établissement d'une AR. À la réception d'une primitive "indication" Establish, l'ASE d'AP appelé examine les paramètres

spécifiés dans celle-ci et retourne la réponse appropriée dans la primitive "response" AR-Establish.

#### 6.2.3.3.4.3.2 Établissement d'AR de type un vers plusieurs

Lorsqu'il est utilisé à l'appui des AREP de type un à plusieurs, le service AR-Establish fait que les AREP locaux sont établis de façon indépendante (voir Figure 3).




**Légende**

| Anglais | Français |
|---|---|
| Each AP in the AR establishes its own AREP locally by issuing an Establish request. Establish PDUs are not exchanged.<br>However, if the AR is supported by a data link connection, DL-PDUs are exchanged | Chaque AP dans l'AR établit son propre AREP localement en émettant une demande Establish. Les PDU Establish ne sont pas échangées.<br>Toutefois, si l'AR est prise en charge par une connexion de liaison de données, les PDU de DL sont échangées. |
| request | demande |
| DLSAP | DLSAP |
| AREP | AREP |
| AP | AP |

**Figure 3 – Établissement d'AR de type un vers plusieurs**

À la réception d'une primitive "request" de service AR-Establish, la FAL appelante émet une primitive "confirmation" de service AR-Establish vers l'AP appelant indiquant si, oui ou non, elle était capable d'établir l'AREP. Si l'un des cas suivants est avéré, elle n'est pas capable d'établir l'AREP:

1) l'AREP demandé n'est pas spécifié dans la FAL, ou
2) les ressources ne sont pas disponibles pour établir l'AREP demandé, ou
3) pour les AREP pris en charge par une connexion de liaison de données, la couche liaison de données n'était pas capable d'établir la connexion de liaison de données.

#### 6.2.3.3.4.3.3 Classes d'AREP compatibles

Le Tableau 30 fourni des combinaisons possibles des classes d'AREP qui peuvent être reliées au service AR-Establish. Dans ce tableau, la colonne SERVER contient un "-" pour indiquer les AREP serveurs ne lancent pas d'établissement d'AR.

**Tableau 30 – Combinaisons valides de classes d'AREP à relier**

| | | Rôle de l'AREP appelant | | |
|---|---|---|---|---|
| | | **PEER** | **CLIENT** | **SERVER** |
| Appelé | PEER | OUI | OUI | -- |
| AREP | CLIENT | N° | N° | -- |
| Role | SERVER | OUI | OUI | -- |

Si l'AP appelé décide d'établir une AR, il émet une primitive "response" AR-Establish. La FAL retourne le paramètre AR qui est utilisé pour se référer à l'AR formée à partir de l'utilisateur de service appelé. Si l'AR en établissement utilise la couche liaison de données orientée connexion et son établissement explicite est requis, il est établi avant qu'une APDU de réponse Establish ne soit retournée. L'AP appelé retourne ensuite une APDU de réponse Establish avec le paramètre Result(+) à l'AR_ASE appelant. L'AR_ASE appelant émet une primitive "confirmation" AR-Establish dans laquelle le paramètre d'AR est spécifié.

#### 6.2.3.3.4.3.4 Résolution de conflit

L'établissement normal des AR suit la relation séquence-temps générale pour des services confirmés. Lorsque chaque point d'extrémité d'une AR émet en même temps une demande AR-Establish, il se produit un conflit. Les algorithmes de résolution de conflits de ce type sont spécifiés dans le détail dans la CEI 61158-6-5.

### 6.2.3.3.5 Service AR-deEstablish

#### 6.2.3.3.5.1 Vue d'ensemble du service

Ce service confirmé est invoqué par les utilisateurs clients ou homologues pour demander l'arrêt progressif d'une relation entre applications de type un vers un. L'utilisation du service AR-DeEstablish entraîne la fermeture des points d'extrémité de l'AR. Ils peuvent être rouverts à l'aide du service AR-Establish. Ce service peut être utilisé sur n'importe quel AREP ouvert, que l'AR ait été préétablie ou établie de façon dynamique à l'aide du service AR-Establish.

Lorsqu'une demande de désinstallation une AR est faite, l'ASE d'AR local achemine la PDU de demande DeEstablish vers le point d'extrémité distant de l'AR et n'accepte aucune autre PDU de demande provenant de l'utilisateur, sauf si un service AR-Abort est demandé. À la réception d'une PDU de DeEstablish, il efface toutes les demandes de service en cours, ferme le contexte des points d'extrémité et informe l'utilisateur en utilisant la primitive de service "confirm".

L'ASE d'AR distant informe son utilisateur lorsqu'une PDU de demande AR-DeEstablish est reçue en utilisant la primitive "indication" pour le service. L'utilisateur retourne une primitive "response" après avoir répondu aux indications de service qu'il a reçues. Lorsque l'ASE d'AR reçoit la réponse, il ferme le contexte des points d'extrémité et retourne une PDU de réponse AR-DeEstablish.

#### 6.2.3.3.5.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 31. L'utilisation de ce service est l'objet d'une étude future.

**Tableau 31 – Service AR-DeEstablish**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du 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.

#### 6.2.3.3.5.3 Procédure du service

Le service "AR-DeEstablish" est un service qui opère à travers une file d'attente ou un buffer. Sa procédure de service est l'objet d'une étude future.

### 6.2.3.3.6 Service AR-abort

#### 6.2.3.3.6.1 Vue d'ensemble du service

Le service est utilisé par l'utilisateur de la FAL pour mettre brutalement fin à une AR. Il réussit toujours; le destinataire d'une demande Abort ou d'une APDU de demande Abort abandonne toujours l'AR. Ce service peut être utilisé sur n'importe quel AREP ouvert, que l'AR ait été préétablie ou établie de façon dynamique à l'aide du service Establish.

Le service AR-Abort est utilisé pour donner à l'ASE de l'AR l'instruction de mettre brutalement fin à toute l'activité sur un AREP et de le mettre à l'état Closed (fermé). La réception d'une primitive "request" de service AR-Abort amène l'ASE de l'AR à fermer immédiatement le contexte de l'AREP et à émettre une APDU de demande Abort vers les AREP distants. La réception d'une APDU de demande Abort amène l'ASE de l'AR à fermer immédiatement le contexte de l'AREP de l'AR et à renvoyer à l'utilisateur une primitive "indication" de demande AR-Abort. La fermeture immédiate de chaque contexte de point d'extrémité entraîne la suppression de toutes les demandes de service en cours. Toutes les primitives de service et APDU ultérieures reçues par l'ASE d'AR pour l'AR abandonnée sont rejetées, à l'exception de celles du service AR-Establish.

Le service AR-Abort peut être demandé au niveau de l'AREP d'une relation de type un vers un ou bien au niveau du point d'extrémité éditeur d'une AR de type un vers plusieurs ou du type un vers un. Les abonnés ne sont pas capables d'abandonner des AR avec éditeurs, bien qu'ils puissent fermer leurs propres points d'extrémité par des moyens locaux.

L'ASE d'AR peut également lancer le service Abort lorsqu'il détecte des défaillances de communication irrécupérables. Dans ce cas, l'ASE de l'AR livre une primitive "indication" AR-Abort informant l'utilisateur de la défaillance et ferme le contexte des points d'extrémité.

#### 6.2.3.3.6.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 32.

**Tableau 32 – AR-Abort**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Locally Generated | | M |
| Originator | M | M (=) |
| Reason Code | M | M (=) |
| Additional Detail | U | U (=) |

**Argument**

Ce paramètre contient l'information associée au service AR-Abort.

**Locally Generated**

Ce paramètre spécifie si, oui ou non, l'abandon a été généré localement.

**Originator**

Ce paramètre identifie l'origine de l'abandon. Ses valeurs valides sont DLL, FAL, ou FAL-USER. La valeur DLL ne peut pas êtue utilisée dans la primitive "request".

**Reason Code**

Ce paramètre indique la cause de l'abandon. Il peut être fourni par le fournisseur ou par l'utilisateur. Une seule cause est définie: AR ASE Error (Erreur d'ASE d'AR). D'autres valeurs du code de cause peuvent être fournies par la couche liaison de données ou par l'utilisateur.

**Additional Detail**

Ce paramètre facultatif spécifie les données d'utilisateur qui accompagnent l'indication. Lorsqu'il est utilisé, la valeur présentée dans la primitive "request" est livrée inchangée dans la primitive "indication".

#### 6.2.3.3.6.3 Procédure du service

Le service Abort est un service qui opère à travers une file d'attente ou un buffer.

Si l'utilisateur souhaite abandonner une AR, il présente une primitive "request" de service Abort à son ASE de l'AR de la FAL. Si un ASE de l'AR détecte une défaillance locale irrécupérable ou une défaillance de communication, il livre une primitive "indication" de service Abort à l'utilisateur de point d'extrémité.

L'ASE de l'AR de la FAL construit une APDU de demande d'abandon et l'achemine sur l'AR spécifiée s'il est capable d'agir ainsi. Il fait également passer l'état de l'AREP à CLOSED.

A la réception de l'APDU de demande Abort, chaque ASE de l'AR de la FAL fait passer son AREP à CLOSED et livre une primitive "indication" de service Abort à son utilisateur.

#### 6.2.3.3.7 Service AR-compel

##### 6.2.3.3.7.1 Vue d'ensemble du service

Ce service est utilisé par l'utilisateur de la FAL pour demander à l'ASE de l'AR d'acheminer un message qui avait été différé pour être libéré de façon explicite.

NOTE Les services qui opèrent sur des AR sont décrits de manière abstraite afin qu'ils soient capables d'opérer avec des AR qui acheminent des APDU de FAL par des buffers ou par des files d'attente. Ces services peuvent être mis en œuvre d'une manière fournissant la capacité de charger le buffer/la file d'attente et ensuite de l'afficher en vue de son transfert par la couche liaison de données sous-jacente en utilisant ce service.

##### 6.2.3.3.7.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 33.

**Tableau 33 – Service AR-Compel**

| Nom de paramètre | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Schedule ID | U | |
| Result (+) | | S |
| Status | | M |
| Result (-) | | S |
| Error Info | | M |
| NOTE Voir la Note en 3.8.4.3. | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Schedule ID**

Ce paramètre facultatif spécifie la séquence de couche liaison de données devant être forcée pour les AR programmées par un réseau. La séquence de programmation fait partie du mapping de DL défini dans la CEI 61158-6-5.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Status**

Ce paramètre indique le résultat de la demande de service. Les trois codes de statut suivants fournis par la DLL sont définis

success (succès);

failure – inappropriate request (échec – demande inappropriée);

failure – reason unspecified (échec – cause non spécifiée).

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

##### 6.2.3.3.7.3 Procédure du service

Le service "AR-Compel" est un service qui opère à travers une file d'attente ou un buffer.

L'utilisateur demandeur présente à son ASE d'AR de FAL une primitive "request" de service AR compel. L'ASE de l'AR de FAL émet la demande de service de couche liaison de données correspondante vers la couche liaison de données.

#### 6.2.3.3.8 Service AR-get buffered message

##### 6.2.3.3.8.1 Vue d'ensemble du service

Ce service local est utilisé par un processus application pour demander à l'ASE de l'AR de récupérer un message qui est actuellement maintenu dans un buffer de la couche liaison de données locale.

Ce service ne conduit pas à l'acheminement d'une APDU. Il est fourni afin que l'utilisateur de la FAL puisse accéder à un buffer par l'AR de FAL.

##### 6.2.3.3.8.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 34.

**Tableau 34 – Service Service AR-Get buffered message**

| Nom de paramètre | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| | | |
| Result(+) | | S |
| Decoded buffer data | | M |
| Local timeliness | | C |
| Remote timeliness | | C |
| Duplicate FAL PDU Body | | C |
| | | |
| Result (-) | | S |
| Error Info | | M |
| NOTE Voir la Note en 3.8.4.3. | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Decoded Buffer Data**

Ce paramètre spécifie les données d'utilisateur dans l'APDU de FAL lue dans le buffer.

**Local Timeliness**

Ce paramètre conditionnel, s'il est présent et True (vrai), indique que le paramètre Decoded Buffer Data a satisfait aux critères de cohérence temporelle de réception définis pour la DLL. Si sa valeur est False, au moins l'un des critères de cohérence temporelle n'a pas été respecté. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

**Remote Timeliness**

Ce paramètre conditionnel, s'il est présent et True (vrai), indique que le paramètre Decoded Buffer Data a satisfait aux critères de cohérence temporelle de l'éditeur et de la DL émettrice. Si sa valeur est False, au moins l'un des critères de cohérence temporelle n'a pas été

respecté. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

**Duplicate FAL PDU Body**

Ce paramètre conditionnel indique si, oui ou non, la réception d'une PDU de FAL dupliquée a été détectée par la couche liaison de données. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.3.3.8.3 Procédure du service

Ce service demande à la FAL de retourner le contenu actuel du buffer dans une primitive confirmation(+). Si le buffer est vide, une primitive confirmation(-) est retournée.

### 6.2.3.3.9 Service AR-schedule communication

#### 6.2.3.3.9.1 Vue d'ensemble du service

Ce service local donne à l'utilisateur de l'AL la capacité de programmer une séquence de DL-COMPEL pour une relation particulière. Il mappe directement le service DL-Schedule-Sequence et n'a aucun effet sur les diagrammes d'états de l'AR.

Ce service ne conduit pas à l'acheminement d'une APDU. Il peut, toutefois, conduire au transfert des PDU de couche liaison de données.

#### 6.2.3.3.9.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 35.

**Tableau 35 – Service AR-Schedule communication**

| Nom de paramètre | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Invoke ID | U | |
| Schedule Information | M | |
| | | |
| Result (+) | | S |
| Invoke ID | | U (=) |
| Sequence ID | | M |
| | | |
| Result (-) | | S |
| Invoke ID | | U (=) |
| Error Info | | M |
| NOTE Voir la Note en 3.8.4.3. | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Schedule Info**

Ce paramètre spécifie les informations de programmation de couche liaison de données pour la communication programmée.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Sequence ID**

Ce paramètre fournit un identificateur abrégé pour la communication programmée demandée.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.3.3.9.3 Procédure du service

Ce service demande à la pile communication de créer une séquence programmée.

### 6.2.3.3.10 Service AR-cancel scheduled sequence

#### 6.2.3.3.10.1 Vue d'ensemble du service

Ce service local donne à l'utilisateur de l'AL la capacité d'annuler une séquence existante qui avait été programmée précédemment. Il mappe directement au service DL-Cancel-Schedule et n'a aucun effet sur les diagrammes d'états d'AR.

Ce service ne conduit pas à l'acheminement d'une APDU. Il peut, toutefois, conduire à l'acheminent des PDU de couche liaison de données.

#### 6.2.3.3.10.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 36.

**Tableau 36 – Service AR-Cancel scheduled sequence**

| Nom de paramètre | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Invoke ID | U | |
| Sequence ID | M | |
| | | |
| Result (+) | | S |
| Invoke ID | | U (=) |
| | | |
| Result (-) | | S |
| Invoke ID | | U (=) |
| Error Info | | M |
| NOTE Voir la Note en 3.8.4.3. | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Sequence ID**

Ce paramètre spécifie le programme de la séquence devant être annulée.

**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.

#### 6.2.3.3.10.3 Procédure du service

Ce service demande à la pile communication d'annuler une séquence programmée existante.

### 6.2.3.3.11 Service AR-get DL-time

#### 6.2.3.3.11.1 Description

Ce service est défini pour la FAL pour donner à l'utilisateur de la FAL un accès au service DL-Time de la couche liaison de données. Afin de maintenir la compatibilité avec le service DL-Time, les définitions de service et des paramètres ne sont pas définies dans la présente norme. Voir la CEI 61158-3-1 pour la description du service DL-Time.

### 6.2.3.3.12 Service AR-status

#### 6.2.3.3.12.1 Vue d'ensemble du service

Ce service local fournit à l'utilisateur de l'AL la notification d'un changement de statut de l'AREP.

#### 6.2.3.3.12.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 37.

**Tableau 37 – AR-Status**

| Nom de paramètre | Ind |
|---|---|
| Argument | |
| AREP | M |
| Status code | M |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Status Code**

Ce paramètre spécifie le changement de statut rapporté. Les codes de statut suivants sont définis

- lower layer reset (réinitialisation de couche inférieure);
- buffer received (buffer reçu);
- buffer transmitted (buffer émis);
- transmission not timely (émission non opportune);
- lower layer lost schedule – rescheduling required (perte de programme de couche inférieure – reprogrammation requise);
- local confirmation (confirmation locale).

#### 6.2.3.3.12.3 Procédure du service

Ce service indique qu'un événement significatif, tel que défini par le paramètre status code (Code de statut), s'est produit dans la pile communication.

#### 6.2.3.3.13 Service AR-XON-OFF

##### 6.2.3.3.13.1 Vue d'ensemble du service

Ce service local entraîne la suspension ou la reprise du flux de données sur une AR spécifiée.

##### 6.2.3.3.13.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 38.

**Tableau 38 – AR-XON-OFF**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| XON-OFF | M | M (=) |

**Argument**

L'argument contient les paramètres de la demande du service.

**XON-OFF**

Ce paramètre spécifie les données fournies par un utilisateur qui peuvent être acheminées avec la demande de service. Les données d'utilisateur peuvent être "ON“ ou "OFF“.

##### 6.2.3.3.13.3 Procédure du service

Le service "AR-XON-OFF" est un service qui opère à travers une file d'attente.

L'ASE de FAL demandeur présente à son ASE d'AR une primitive "request" de service AR-XON-OFF. L'ASE d'AR construit une APDU de demande de service AR-XON-OFF.

NOTE Le programme d'émission est géré par la couche sous-jacente, pas par l'ASE d'AR. Se référer à la CEI 61158-3-1 et à la CEI 61158-4-1 pour de plus amples détails.

À la réception de l'APDU de demande de service AR-XON-OFF, l'ASE d'AR destinataire livre une primitive AR-XON-OFF.indication à l'ASE de FAL approprié comme indiqué dans le paramètre FAL Service Type.

#### 6.2.3.3.14 Service AR remote read

##### 6.2.3.3.14.1 Vue d'ensemble du service

Ce service fournit le processus application pour demander à l'ASE d'AR de récupérer un message qui est actuellement maintenu dans un buffer de la couche liaison de données distante.

##### 6.2.3.3.14.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 39.

**Tableau 39 – Service AR-Remote read**

| Nom de paramètre | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Priority | M | |
| | | |
| Result (+) | | S |
| Decoded Buffer Data | | M |
| Local Timeliness | | C |
| Remote Timeliness | | C |
| | | |
| Result (-) | | S |
| Error info | | M |
| NOTE Voir la Note en 3.8.4.3. | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Priority**

Cet argument est utilisé et défini localement par l'utilisateur pour permettre deux flux de données.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Decoded Buffer Data**

Ce paramètre spécifie les données d'utilisateur dans l'APDU de FAL lue dans le buffer.

**Local timeliness**

Ce paramètre conditionnel, s'il est présent et True (vrai), indique que le paramètre Decoded Buffer Data a satisfait aux critères de cohérence temporelle de réception définis pour la DLL. Si sa valeur est False, au moins l'un des critères de cohérence temporelle n'a pas été respecté. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

**Remote timeliness**

Ce paramètre conditionnel, s'il est présent et True (vrai), indique que le paramètre Decoded Buffer Data a satisfait aux critères de cohérence temporelle de l'éditeur et de la DL émettrice. Si sa valeur est False, au moins l'un des critères de cohérence temporelle n'a pas été respecté. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.3.3.14.3 Procédure du service

Ce service demande à la FAL de retourner le contenu actuel du buffer distant et de donner la valeur du contenu dans une primitive confirmation(+). Si le buffer est vide, une primitive confirmation(-) est retournée.

#### 6.2.3.3.15 Service AR-remote write

##### 6.2.3.3.15.1 Vue d'ensemble du service

Ce service donne à l'utilisateur de l'AL la capacité de déclencher un échange de données, après avoir écrit une valeur dans un buffer local.

##### 6.2.3.3.15.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 40.

**Tableau 40 – Service AR-Remote write**

| Nom de paramètre | Req | Cnf |
|---|---|---|
| Argument | | |
| AREP | M | |
| Priority | M | |
| Decoded Buffer Data | M | |
| Result (+) | | S |
| Result (-) | | S |
| Error Info | | M |
| NOTE Voir la Note en 3.8.4.3. | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Priority**

Cet argument est utilisé et défini localement par l'utilisateur pour permettre deux flux de données.

**Decoded Buffer Data**

Cet argument est utilisé pour contenir l'APDU devant être écrite.

**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.

#### 6.2.3.4 Procédure du service

Ce service demande à la FAL d'écrire le contenu du buffer local et de déclencher le transfert de données. Une primitive confirmation(+) est envoyée lorsque le service réussit. Si le service échoue, une primitive confirmation(-) est retournée.

### 6.2.4 ASE Variable

#### 6.2.4.1 Vue d'ensemble

Dans l'environnement de bus de terrain, les processus application contiennent des données que des applications distantes sont capables de lire et écrire. L'ASE Variable définit des attributs, visibles du réseau, des données application et fournit un jeu de services utilisés pour lire, écrire et rapporter leurs valeurs. Des services de gestion de FAL communs sont utilisés pour créer et supprimer des objets variables et pour accéder à leurs attributs.

Deux classes de variables d'APO sont définies par le modèle de variable, à savoir les variables individuelles et les listes de variables. Les variables individuelles sont utilisées pour accéder à des données application de n'importe quel type de données défini par la FAL ou défini par un utilisateur. Les services définis pour accéder à des variables individuelles peuvent être utilisés pour accéder à une ou plusieurs variables complètes ou à une ou plusieurs entrées (par variable) dans une matrice ou à un ou plusieurs champs (par variable) dans une structure.

Les listes de variables sont utilisées pour regrouper des variables pour des besoins d'accès. Les mêmes services utilisés pour accéder à des variables individuelles peuvent être utilisés pour accéder à une liste de variables.

Deux types d'opérations sont pris en charge par les services de variables, à savoir "best effort" et "atomic". Les services de "best effort" réussissent lorsqu'au moins l'une des valeurs référencées est accessible. Les services "atomic" réussissent lorsque toutes les valeurs référencées sont accessibles, ils échouent si l'une quelconque des valeurs n'est pas accessible.

Lorsqu'il est pris en charge par le type approprié de relation entre applications, le service Variable Model peut être utilisé pour prendre en charge deux différents modèles d'accès, à savoir le modèle client/serveur et le modèle éditeur/abonné. Le modèle client/serveur est caractérisé en ce qu'une application Client envoie une demande de lecture ou d'écriture à une application Serveur qui répond en conséquence. L'activité du serveur est stimulée par les clients sur le réseau. S'il n'y a pas de demande, le serveur ne génère pas de réponse.

Le modèle éditeur/abonné est différent. Il est caractérisé en ce qu'un éditeur de données édite ses données sur le réseau. Les abonnés souhaitant acquérir les données produites rejoignent la relation entre applications utilisée pour les éditer et être à l'écoute des données à mesure qu'elles sont émises.

Deux modèles sont fournis pour prendre en charge cette activité éditeur/abonné, à savoir le "pull" (tirage) et le "push" (poussée). Dans le modèle "pull", le gestionnaire d'édition tire les données issues de l'éditeur en lui émettant une demande de lecture. L'éditeur répond en diffusant par multidiffusion une séquence de réponses de lecture vers le gestionnaire d'édition et vers les abonnés.

Dans le modèle "push", le gestionnaire d'édition est toujours copositionné avec l'éditeur. Les abonnés sont capables d'indiquer au gestionnaire d'édition comment ils souhaitent que les données soient produites. Par des interfaces locales, le gestionnaire d'édition commande l'activité de l'éditeur et les caractéristiques des relations entre applications utilisées pour distribuer les données. Dans ce cas, les données produites sont émises sur le réseau en utilisant le service non confirmé Information Report.

Les abonnés dans les deux modèles reçoivent les données produites par le biais d'une copie locale des données maintenues par la FAL. À mesure que les données sont reçues du réseau, la copie locale est mise à jour et rendue disponible pour l'abonné. Lorsque l'abonné souhaite accéder aux données, il accède à la copie locale, au lieu d'émettre une demande de lecture vers la variable distante comme cela se produirait dans le modèle client/serveur. Si les attributs d'AREP appropriés sont mis, les informations relatives à la cohérence temporelle seront incluses avec les données.

L'ASE d'AR de la FAL prend en charge les deux modèles éditeur/abonné par multidiffusion des données produites sur les relations entre applications programmées par un réseau et mises en buffer. Les caractéristiques exactes des transferts sont commandées par les valeurs de réglage d'attributs de la relation entre applications.

Le modèle formel du modèle de variables est présenté ci-après, suivi d'une description de ses services. La CEI 61158-6-5 décrit la syntaxe abstraite et les procédures pour son protocole.

#### 6.2.4.2 Exigences relatives à l'accès aux variables

La structure d'une valeur variable est définie par son data type. Un data type de variable peut être de n'importe quel type de données valide normalisé ou défini par un utilisateur. Les types normalisés et les moyens pour définir les types d'utilisateur sont spécifiés par le modèle de Data type.

Le modèle Data type permet l'imbrication de structures de données et de matrices à un plus d'un niveau et, de ce fait, les services définis pour l'accès aux variables permettent un accès partiel à n'importe quel niveau d'imbrication. «Accès partiel» signifie accès indépendant à plusieurs composants de chaque structure ou matrice référencée. Autrement dit, si le type construit "A" contient le type construit "B", l'accès partiel est fourni à "B" et aussi aux composants de "B".

#### 6.2.4.3 Spécification de la classe-modèle "Variable"

##### 6.2.4.3.1 Spécification de la classe "Simple Variable" (Variable simple)

###### 6.2.4.3.1.1 Modèle formel de Simple Variable

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE VARIABLE** |
| **CLASS:** | | **SIMPLE VARIABLE** | |
| **CLASS ID:** | | | 7 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (o) | Attribut clé: | Symbolic Address |
| 2 | (m) | Attribut: | Data type |
| 3 | (m) | Attribut: | Length |
| 4 | (c) | Contrainte: | Data type Format = STRING |
| 4.1 | (o) | Attribut: | Variable Length Conveyance (TRUE, FALSE) -- voir Note ci-dessous |
| 5 | (m) | Attribut: | Access Privilege |
| 5.1 | (m) | Attribut: | Password |
| 5.2 | (m) | Attribut: | Access Groups |
| 5.3 | (m) | Attribut: | Access Rights |
| 6 | (m) | Attribut: | Local Detail |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Information Report |
| 6 | (o) | OpsService: | Information Report List |
| 7 | (o) | OpsService: | Exchange |
| 8 | (o) | OpsService: | Exchange List |

NOTE La contrainte est TRUE lorsqu'une variable est définie comme une matrice.

###### 6.2.4.3.1.2 Attributs

**Symbolic Address**

Cet attribut-clé facultatif spécifie une référence symbolique pour la variable. Cet attribut fournit un second espace de nom pour définir des variables qui est distinct de celui du nom de variable pour assurer que la duplication de noms dans des espaces de nom distincts ne crée pas de problème. L'adresse symbolique peut être utilisée pour affecter des noms de variable indépendants de la manière dont ils sont nommés dans un système donné.

Il n'est pas permis d'intégrer un caractère espace (" ") au milieu d'une adresse symbolique. Par contre, il est permis de l'utiliser comme caractères de remplissage de tête ou de queue.

De tels caractères de remplissage ne sont pas à interpréter comme faisant partie de l'adresse symbolique. L'utilisation des caractères de remplissage de tête ou de queue permet l'emploi de chaînes de caractères de longueur fixe.

NOTE Par exemple, un fabricant d'appareil peut utiliser l'adresse symbolique pour nommer les données acquises provenant d'une voie d'entrée locale comme étant "port 1". Un utilisateur de ces données peut créer une seconde variable pour les données en utilisant le nom de variable "température".

**Data type**

Cet attribut est l'identificateur numérique d'un data type FIXED-LENGTH ou STRING.

**Length**

Cet attribut indique la longueur des data types simple variable. Pour les variables avec des data types STRING, cet attribut est utilisé pour définir la longueur maximale de la variable en octets. Pour les variables avec le data type FIXED-LENGTH, cet attribut est utilisé pour refléter la longueur de la variable telle que spécifiée par le data type.

**Variable Length Conveyance**

Cet attribut Boolean conditionnel, lorsqu'il est TRUE, indique que seuls les octets valides de la chaîne sont acheminés. Lorsqu'il est FALSE, tous les octets de la chaîne sont acheminés, même s'ils ne font pas partie d'une valeur string. Cet attribut est présent seulement pour les variables avec le data type STRING.

**Access Privilege**

Cet attribut spécifie les contrôles d'accès définis pour cette variable. Il se compose des éléments suivants:

**Password**

Cet attribut spécifie le mot de passe pour les droits d'accès. Sa valeur est null s'il n'est pas utilisé.

**Access Groups**

Cet attribut identifie lesquels des huit groupes d'accès définis par un utilisateur sont définis pour la variable. Plus d'un groupe d'accès peut être défini.

**Access Rights**

Cet attribut définit le type d'accès défini pour la variable. Les valeurs valides sont:

- Droit d'écrire pour les groupes d'accès
- Droit de lire pour les groupes d'accès
- Droit d'écrire pour le mot de passe enregistré
- Droit de lire pour le mot de passe enregistré
- Droit d'écrire pour tous les partenaires de communication
- Droit de lire pour tous les partenaires de communication

**Local Detail**

Cet attribut spécifie des informations locales.

#### 6.2.4.3.1.3 Services

Tous les services définis pour cette classe sont facultatifs. Lorsqu'une instance de la classe est définie, l'un au moins de ces services est à prendre en charge.

**Read**

Ce service facultatif peut être utilisé pour lire un seul objet variable ou objet variable list. Ce service peut être utilisé tant dans le modèle client/serveur que dans le modèle "pull" éditeur/abonné.

**Write**

Ce service facultatif peut être utilisé pour mettre à jour un seul objet variable ou objet variable list. Ce service ne peut être utilisé que dans le modèle client/serveur.

**Read List**

Ce service facultatif peut être utilisé pour lire plusieurs objets "variable" ou composants de plusieurs objets "variable" ou un seul objet "variable list". Le nombre de variables/composants ou d'objets "variable" référencés dans la liste relève de l'initiative du constructeur d'appareil. Chaque variable/composant ou variable dans la liste référencé(e) est accessible et retourné(e) en "best effort", signifiant qu'au moins une valeur est retournée pour que le service réussisse. Ce service ne peut être utilisé que dans le modèle client/serveur.

**Write List**

Ce service facultatif peut être utilisé pour mettre à jour plusieurs objets "variable" ou composants individuels de plusieurs objets "variable" ou un seul objet "variable list". Le nombre de variables/composants ou d'objets "variable" référencés dans la liste relève de l'initiative du constructeur d'appareil. Chaque variable/composant ou variable dans la liste référencé(e) est mis(e) à jour et retourné(e) en "best effort", signifiant qu'au moins une valeur est mise à jour pour que le service réussisse. Ce service ne peut être utilisé que dans le modèle client/serveur.

**Information Report**

Ce service facultatif est un service non confirmé qui peut être utilisé pour rapporter la valeur d'un objet "variable" ou "variable list". Ce service peut être utilisé tant dans le modèle client/serveur que dans le modèle "push" éditeur/abonné.

**Information Report List**

Ce service facultatif non confirmé peut être utilisé pour rapporter plusieurs objets "variable" ou composants individuels de plusieurs objets "variable" ou un seul objet "variable list". Le nombre de variables/composants ou d'objets "variable" référencés dans la liste relève de l'initiative du constructeur d'appareil. Ce service peut être utilisé tant dans le modèle client/serveur que dans le modèle "push" éditeur/abonné.

**Exchange**

Ce service facultatif est un service confirmé qui peut être utilisé pour écrire la valeur d'une variable ou liste de variables distante et lire la valeur d'une autre variable ou liste de variables en une seule et même opération. Ce service peut être utilisé dans le modèle client/serveur. La relation entre les variables spécifiées et une fonction d'échange de la couche utilisateur ne relève pas du domaine d'application de la présente norme.

**Exchange List**

Ce service confirmé facultatif peut être utilisé pour mettre à jour plusieurs objets "variable" ou composants individuels de plusieurs objets "variable" ou un seul objet "variable list" et pour lire plusieurs objets "variable" ou composants individuels de plusieurs objets "variable" ou un seul objet " variable list" en une seule et même opération. Le nombre de variables/composants ou d'objets "variable" référencés dans la liste relève de l'initiative du constructeur d'appareil. Chaque variable/composant ou variable dans la liste référencé(e) est mis(e) à jour/accessible en "best effort", signifiant qu''au moins une valeur est mise à jour/retournée pour que le service réussisse. Ce service ne peut être utilisé que dans le modèle client/serveur. La relation entre les variables spécifiées et une fonction d'échange de la couche utilisateur ne relève pas du domaine d'application de la présente norme.

#### 6.2.4.3.2 Spécification de la classe "Array variable" (variable matrice)

##### 6.2.4.3.2.1 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE VARIABLE** |
| **CLASS:** | | **ARRAY VARIABLE** | |
| **CLASS ID:** | | | 8 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (o) | Attribut clé: | Symbolic Address |
| 2 | (m) | Attribut: | Data type (Type de données) |
| 2.1 | (s) | Attribut: | ID de type de donnée |
| 2.2 | (s) | Attribut: | Embedded Data type |
| 3 | (c) | Contrainte: | Data type Format = ARRAY \| FIXED-LENGTH \| STRING |
| 3.1 | (o) | Attribut: | Element Length |
| 4 | (c) | Contrainte: | Data type Format = STRING |
| 4.1 | (o) | Attribut: | Variable Length Conveyance (TRUE, FALSE) |
| 5 | (o) | Attribut: | Number of Elements |
| 6 | (m) | Attribut: | Access Privilege |
| 6.1 | (m) | Attribut: | Password |
| 6.2 | (m) | Attribut: | Access Groups |
| 6.3 | (m) | Attribut: | Access Rights |
| 7 | (m) | Attribut: | Local Detail |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Information Report |
| 6 | (o) | OpsService: | Information Report List |
| 7 | (o) | OpsService: | Exchange |
| 8 | (o) | OpsService: | Exchange List |

##### 6.2.4.3.2.2 Attributs

**Symbolic Address**

Voir la description de cet attribut défini pour la classe Simple Variable ci-dessus.

**Data type**

Cet attribut conditionnel spécifie le type de donnée pour les éléments de la matrice.

**Data type ID**

Cet attribut est l'identificateur numérique d'un data type associé à la matrice. Si le format du data type est ARRAY, cet attribut est le data type de la matrice. Si le data type est FIXED-LENGTH ou STRING, cet attribut est le data type d'un élément de matrice.

**Embedded Data type**

Cet attribut est utilisé pour intégrer la description de data type dans la définition de la matrice. Le contenu pour cet attribut est montré dans le modèle Data type.

**Element Length**

Cet attribut conditionnel indique la longueur d'un élément de matrice ayant le data type ARRAY, FIXED-LENGTH ou STRING. Pour les éléments de matrice avec des data types STRING, cet attribut est utilisé pour définir la longueur d'un élément de matrice en octets. Pour les variables avec un data type ARRAY ou FIXED-LENGTH, cet attribut est utilisé pour refléter la longueur d'un élément de matrice tel que spécifié par le data type.

**Variable-length Conveyance**

Cet attribut Boolean conditionnel, lorsqu'il est TRUE, indique que seuls les octets valides de la chaîne sont acheminés. Lorsqu'il est FALSE, tous les octets de la chaîne sont acheminés, même s'ils ne font pas partie d'une valeur string. Cet attribut est présent seulement pour les matrices avec un data type STRING.

**Number of Elements**

Cet attribut spécifie le nombre d'éléments de matrice pour les objets "variable" qui sont définis comme des matrices. Les objets Array Variable peuvent être définis de l'une de deux façons. Chacune des façons et l'utilisation de cet attribut pour chacune sont montrées ci-dessous.

| Méthode | Utilisation de l'attribut |
|---|---|
| Référence à un data type avec le format ARRAY: | Reflète le nombre d'éléments définis pour le data type "array" |
| Référence à un data type avec le format FIXED-LENGTH ou STRING | Spécifie le nombre d'éléments définis pour la variable de matrice. |

**Access Privilege**

Voir la description de cet attribut et de ses composants définis dans la classe Simple Variable ci-dessus.

**Local Detail**

Cet attribut spécifie des informations locales.

#### 6.2.4.3.2.3 Services

Voir la description des services définis dans la classe Simple Variable ci-dessus.

### 6.2.4.3.3 Spécification de la classe "Record Variable" (variable enregistrement)

#### 6.2.4.3.3.1 Modèle formel

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

| | | | |
|---|---|---|---|
| 1 | (o) | Attribut clé: | Symbolic Address |
| 2 | (m) | Attribut: | Data type |
| 2.1 | (s) | Attribut: | Data type ID |
| 2.2 | (s) | Attribut: | Embedded Data type |
| 3 | (m) | Attribut: | Access Privilege |
| 3.1 | (m) | Attribut: | Password |
| 3.2 | (m) | Attribut: | Access Groups |
| 3.3 | (m) | Attribut: | Access Rights |
| 4 | (m) | Attribut: | List of Local Detail |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Information Report |
| 6 | (o) | OpsService: | Information Report List |
| 7 | (o) | OpsService: | Exchange |
| 8 | (o) | OpsService: | Exchange List |

#### 6.2.4.3.3.2 Attributs

**Symbolic Address**

Voir la description de cet attribut défini pour la classe Simple Variable ci-dessus.

**Data type**

Cet attribut conditionnel spécifie le type de donnée pour les éléments de la matrice.

**Data type ID**

Cet attribut est l'identificateur numérique du type de données STRUCTURE.

**Embedded Data type**

Cet attribut est utilisé pour intégrer la description de data type dans la définition de l'enregistrement. Le contenu pour cet attribut est montré dans le modèle Data type.

**Access Privilege**

Voir la description de cet attribut défini pour la classe Simple Variable ci-dessus.

**Password**

Voir la description de cet attribut défini pour la classe Simple Variable ci-dessus.

**Access Groups**

Voir la description de cet attribut défini pour la classe Simple Variable ci-dessus.

**Access Rights**

Voir la description de cet attribut défini pour la classe Simple Variable ci-dessus.

**List of Local Detail**

Cet attribut spécifie le détail local pour chaque élément (champ) de l'enregistrement.

#### 6.2.4.3.3.3 Services

Voir la description des services définis pour la classe Simple Variable ci-dessus.

### 6.2.4.3.4 Spécification de la classe "Variable List" (liste de variables)

#### 6.2.4.3.4.1 Modèle formel

**FAL ASE:** **ASE VARIABLE**
**CLASS:** **VARIABLE LIST**
**CLASS ID:** **10**
**PARENT CLASS:** **TOP**
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | (m) | Attribut: | Number of Entries |
| 2 | (m) | Attribut: | List Of Variables |
| 3 | (c) | Attribut: | Deletable |
| 4 | (m) | Attribut: | Access Privilege |
| 4.1 | (m) | Attribut: | Password |
| 4.2 | (m) | Attribut: | Access Groups |
| 4.3 | (m) | Attribut: | Access Rights |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Read |
| 2 | (o) | OpsService: | Write |
| 3 | (o) | OpsService: | Read List |
| 4 | (o) | OpsService: | Write List |
| 5 | (o) | OpsService: | Exchange |

| | | | |
|---|---|---|---|
| 6 | (o) | OpsService: | Exchange List |
| 7 | (o) | OpsService: | Information Report |
| 8 | (o) | OpsService: | Information Report List |

#### 6.2.4.3.4.2 Attributs

**Number of Entries**

Cet attribut spécifie le nombre de variables dans la liste.

**List of Variables**

Cet attribut identifie les variables (seulement les objets variables simples, matrices ou enregistrements) par l'attribut-clé (Key Attribute) qui sont contenues dans la liste.

**Deletable**

Cet attribut indique, lorsqu'il est TRUE, que la liste de variables peut être effacée au moyen du service Delete. La valeur de cet attribut est toujours TRUE pour les objets créés dynamiquement avec le service Create de l'ASE Object Management.

**Access Privilege**

Cet attribut spécifie les contrôles d'accès définis pour cette liste de variables. Il se compose des éléments suivants:

**Password**

Cet attribut spécifie le mot de passe pour les droits d'accès. Sa valeur est null s'il n'est pas utilisé.

**Access Groups**

Cet attribut identifie lesquels des huit groupes d'accès définis par un utilisateur sont définis pour la liste de variables. Plus d'un groupe d'accès peut être défini.

**Access Rights**

Cet attribut définit le type d'accès défini pour la liste de variables. Les valeurs valides sont:

Droit de supprimer les groupes d'accès

Droit d'écrire les groupes d'accès

Droit de lire les groupes d'accès

Droit de supprimer le mot de passe enregistré

Droit d'écrire le mot de passe enregistré

Droit de lire le mot de passe enregistré

Droit de supprimer tous les partenaires de communication

Droit d'écrire tous les partenaires de communication

Droit de lire tous les partenaires de communication

#### 6.2.4.3.4.3 Services

Voir la description des services définis pour la classe Simple Variable ci-dessus.

### 6.2.4.4 Spécification de services pour l'ASE Variable

#### 6.2.4.4.1 Services pris en charge

Ce paragraphe contient la définition de services qui sont propres à cet ASE. Les services définis pour cet ASE sont:

Read

Write

Read List

Write List

Information Report

Information Report List

Exchange

Exchange List

La nature exacte du fonctionnement des services variables est déterminée, en partie, par la relation entre applications sur laquelle ils opèrent.

#### 6.2.4.4.2 Service "Read"

##### 6.2.4.4.2.1 Vue d'ensemble du service

Ce service confirmé peut être utilisé pour lire la valeur d'un objet variable ou objet variable list. Il peut être utilisé avec des relations entre applications configurées pour prendre en charge le modèle client/serveur ou il peut être utilisé avec des relations entre applications configurées pour prendre en charge le modèle "pull" éditeur/abonné.

##### 6.2.4.4.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 41.

**Tableau 41 – Paramètres du service "Read"**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variable Specifier | M | M (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Data type Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| Best Effort Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Access Results | | | M | M (=) |
| Error Status | | | S | S (=) |
| Returned Data | | | S | S (=) |
| Value | | | M | M (=) |
| Data type | | | C | C (=) |
| Object Revision | | | C | C (=) |
| Timeliness | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE Voir la Note en 3.8.4.3.

**Argument**

Ce paramètre contient les paramètres de l'invocation de service.

**Variable Specifier**

Ce paramètre de type sélecteur spécifie la variable, la liste de variables ou un élément d'une variable matrice ou enregistrement.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Component Identifier**

Ce paramètre identifie le champ d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur de champ d'une structure de données ou d'un indice dans une matrice. Ce paramètre peut identifier un composant avec un niveau d'imbrication supérieur à un, s'il est pris en charge par l'AP.

**Numeric Address/Data Length**

Ce paramètre fournit l'adresse numérique et la longueur de données des données devant être lues. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**Data type Requested**

Ce paramètre facultatif indique qu'il convient que le type de données de chaque variable soit retourné avec les données.

**Object Revision Requested**

Ce paramètre facultatif est utilisé pour demander que la valeur de l'attribut "Object Revision" (révision d'objet) soit retournée. Une valeur TRUE indique que la révision d'objet est souhaitée. Une valeur FALSE indique que la révision d'objet n'est pas souhaitée. Si Object Revision est demandé, mais n'est pas pris en charge pour l'objet spécifié, le service échoue.

**Best Effort Requested**

Ce paramètre facultatif conditionnel est utilisé pour demander d'accomplir une lecture "best effort" d'une liste de variables. Dans la lecture de “best effort”, le service réussit si au moins une variable de la liste de variables peut être lue.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**List of Access Result**

Ce paramètre spécifie les réponses retournées par l'AP distant. Si le “best effort” a été demandé pour une liste de variables, une liste de réponses est retournée. L'ordre des réponses dans la liste est le même que celui des variables dans la liste de variables.

**Error Status**

Ce paramètre de type sélection s'applique lorsque le "best effort" avait été demandé pour une liste de variables et l'une des variables de la liste n'a pas pu être lue. Il indique la cause de l'échec de lecture.

**Returned Data**

Ce paramètre de type sélection spécifie les données retournées par l'AP distant. Il est toujours présent lorsque le "best effort" n'avait pas été demandé pour une liste de variables.

**Value**

Ce paramètre spécifie la valeur lue. Pour chacune des variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans

la liste. Si l'une quelconque des variables de la liste de variables n'a pas pu être lue, le service échoue.

**Data type**

Ce paramètre facultatif indique que le data type de chaque valeur est retourné.

**Object Revision**

Ce paramètre conditionnel spécifie la révision d'objet relative à l'objet spécifié. Il est présent si la révision d'objet a été demandée.

**Timeliness**

Ce paramètre conditionnel indique le statut de cohérence temporelle de couche liaison de données pour l'objet "variable" ou "variable list" référencé. Ce paramètre est présent s'il est pris en charge sur l'AR spécifiée.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.4.4.2.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Read est un service "tout ou rien". "Tout ou rien" signifie que le service réussit seulement si la valeur demandée (ou les valeurs demandées dans le cas d'un objet "variable list") est lue et retournée.

Si l'objet spécifié à lire est un objet "variable list", la valeur, le data type (si demandé) et la révision d'objet (si demandée) des variables dans la liste de variables sont retournés.

Un paramètre de cohérence temporelle est inclus dans la primitive "indication" si l'AR qui a acheminé le corps d'APDU prend en charge la cohérence temporelle.

### 6.2.4.4.3 Service Read list

#### 6.2.4.4.3.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour lire les valeurs de plusieurs variables ou d'un seul objet "variable list". Il peut être utilisé avec des relations entre applications configurées pour prendre en charge le modèle client/serveur.

Il opère en "best effort", signifiant que le service réussit si la valeur pour au moins une variable ou une variable dans une liste de variables est retournée; autrement, il échoue. Lorsque l'accès à la variable a réussi, il convient que la donnée retournée soit la valeur de la variable; autrement, il convient qu'elle soit un "error status" (statut d'erreur).

#### 6.2.4.4.3.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 42.

**Tableau 42 – Paramètres du service Read list**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Data type Requested | U | U (=) | | |
| List Of Variable Specifiers | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID(s) | S | S (=) | | |
| List Of Numeric Addresses and Data Lengths | S | S (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Access Results | | | M | M (=) |
| List of Data types | | | C | C (=) |
| List of Data | | | M | M (=) |
| Error Status | | | S | S (=) |
| Value | | | S | S (=) |
| Value with Object Revision | | | S | S (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE Voir la Note en 3.8.4.3.

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Data type Requested**

Ce paramètre facultatif est utilisé pour demander que la description de type du paramètre List of data soit retournée. Une valeur TRUE indique que la description de type est souhaitée. Une valeur FALSE indique que la description de type n'est pas souhaitée. Cette demande ne peut être utilisée qu'avec la sélection de List of Variable Specifiers.

**List Of Variable Specifiers**

Ce paramètre identifie individuellement une liste de variables et/ou composants de variables devant être lu(e)s ou une seule liste de variables devant être lue. Le niveau d'imbrication des composants devant être lus peut être supérieur ou égal à un.

**Key Attribute**

Ce paramètre de type sélection spécifie les valeurs d'attributs-clés d'identificateurs d'une Variable ou d'une Variable list.

**Key Attribute And Component ID(s)**

Ce paramètre identifie les composants d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur (ou des identificateurs) de chemin des composants.

**List Of Numeric Addresses and Data Lengths**

Ce paramètre fournit l'adresse numérique et la longueur de données des objets "variable" devant être lus. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système

réel relève d'une initiative locale. Cette sélection n'est autorisée à accéder qu'aux données de l'objet variable.

NOTE Les paramètres Numeric Address peuvent être utiles pour spécifier une variable ou un certain emplacement mémoire dans l'utilisateur de service appelé. Ces adresses ne sont pas globales. Un exemple d'utilisation des paramètres consiste à spécifier un pointeur vers un emplacement-cible de la mémoire.

**Object Revision Requested**

Ce paramètre facultatif est utilisé pour demander que la valeur de l'attribut "Object Revision" (révision d'objet) soit retournée. Une valeur TRUE indique que la révision d'objet est souhaitée pour toutes les variables (si prise en charge). Une valeur FALSE indique que la révision d'objet n'est souhaitée pour aucune variable.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**List of Access Results**

Ce paramètre spécifie un indicateur pour chaque variable à laquelle l'accès a eu lieu. Si un accès réussit, il convient que l'indicateur soit TRUE; autrement, il convient qu'il soit FALSE.

**List of Data types**

Ce paramètre facultatif donne la description de type des valeurs dans le paramètre List of Data.

**List of Data**

Ce paramètre spécifie un(e) ou plusieurs statuts d'erreur, valeurs de variable, ou valeurs de variables avec révision d'objet lu(e)s. Les statuts d'erreur ou valeurs avec/sans révision d'objet sont concaténé(e)s dans l'ordre dans lequel ils/elles ont été décrit(e)s dans la demande. Pour les listes de variables, il s'agit toujours de l'ordre de la définition de liste. Pour chaque variable ou une variable dans une liste de variables à laquelle l'accès a réussi, il convient que la donnée retournée soit la valeur ou la valeur avec révision d'objet de l'objet variable (si elle est demandée). Autrement, il convient qu'elle soit un statut d'erreur. Si la révision d'objet est demandée et l'objet variable atteint ne prend pas en charge l'attribut Object Revision, il convient que la donnée retournée soit également un statut d'erreur.

**Error Status**

Ce paramètre indique la cause la plus probable de l'échec de l'opération en utilisant la classe d'erreurs et le code d'erreur définis pour le service de lecture.

**Value**

Ce paramètre spécifie une valeur lue. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Value with Object Revision**

Ce paramètre spécifie une valeur lue plus la révision d'objet de l'objet variable. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées dans l'ordre de leur apparition dans la liste.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.4.4.3.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Read List est un "best effort". "Best effort" signifie que le service réussit si au moins une valeur est lue.

Si l'utilisateur n'est pas capable de lire au moins l'une des valeurs, le service échoue et l'utilisateur produit une primitive "response (-)" du service Read List en indiquant la cause.

Si elles sont demandées, les descriptions de types de données sont retournées. Si elle est demandée, la valeur de l'attribut Objet Revision est retournée en étant concaténée avec la valeur de la variable.

#### 6.2.4.4.4 Service "Write"

##### 6.2.4.4.4.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour écrire la valeur d'une variable. Il n'est pas utilisé dans le modèle éditeur/abonné.

##### 6.2.4.4.4.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 43.

**Tableau 43 – Paramètres du service "Write"**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variable Specifier | M | M (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Value | M | M (=) | | |
| Data type | U | U (=) | | |
| Object Revision | U | U (=) | | |
| Best Effort Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Error Status | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Variable Specifier**

Ce paramètre spécifie la variable, la liste de variables ou un élément d'une variable matrice ou enregistrement.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Component Identifier**

Ce paramètre identifie le champ d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur de champ d'une structure de données ou d'un indice dans une matrice. Ce paramètre peut identifier un composant avec un niveau d'imbrication supérieur à un, s'il est pris en charge par l'AP.

**Numeric Address/Data Length**

Ce paramètre fournit l'adresse numérique et la longueur de données des données devant être écrites. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**Value**

Ce paramètre spécifie la valeur à écrire. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

NOTE Si une variable quelconque d'un objet Variable List ne peut pas être mise à jour, aucune des variables dans l'objet Variable List ne sera mise à jour et l'écriture échouera.

**Data type**

Ce paramètre facultatif spécifie le type de donnés des valeurs dans le paramètre Value.

**Object Revision**

Ce paramètre facultatif spécifie la révision d'objet prévue de l'objet variable. Lorsqu'il est présent, il indique qu'il faut que l'AP contenant la variable compare la valeur de ce paramètre au paramètre Object Revision de la variable avant d'appliquer l'écriture.

**Best Effort Requested**

Ce paramètre facultatif conditionnel est utilisé pour demander d'accomplir une écriture "best effort" d'une liste de variables. Pour l'écriture “best effort”, le service réussit si au moins une variable de la liste de variables peut être écrite.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**List of Status**

Ce paramètre spécifie plus d'un statut si une écriture de “best effort” a été demandée. Le statut indique le succès ou la cause d'échec. L'ordre dans la liste est le même que celui des variables dans la liste de variables.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.4.4.4.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Write est un service "tout ou rien". "Tout ou rien" signifie que le service réussit seulement si la valeur demandée (ou les valeurs demandées dans le cas d'un objet "variable list") est écrite avec succès.

### 6.2.4.4.5 Service Write list

#### 6.2.4.4.5.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour écrire les valeurs de plusieurs variables ou de variables dans un seul objet "variable list". Le nombre de variables référencées ou définies dans la liste

relève d'une initiative locale de l'appareil. Il peut être utilisé avec des relations entre applications configurées pour prendre en charge le modèle client/serveur.

Il opère en "best effort", signifiant que le service réussit si la valeur pour au moins une variable ou une variable dans une liste de variables est écrite; autrement, il échoue.

#### 6.2.4.4.5.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 44.

**Tableau 44 – Paramètres du service Write list**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List of Variable Specifiers | S | S (=)) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component ID(s) | S | S (=) | | |
| List of Numeric Addresses and Data Lengths | S | S (=) | | |
| List of Data types | U | U (=) | | |
| List of Data | M | M (=) | | |
| Value | S | S (=) | | |
| Value with Object Revision | S | S (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Variable Data Access Statuses | | | M | M (=) |
| List of Error Statuses | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante 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 contient les paramètres de l'invocation du service.

**List of Variable Specifiers**

Ce paramètre identifie individuellement plusieurs variables et/ou composants de variables devant être écrit(e)s ou une seule liste de variables devant être écrite. Le niveau d'imbrication des composants devant être écrits peut être supérieur ou égal à un.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Component ID(s)**

Ce paramètre identifie les composants d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur (ou des identificateurs) de chemin des composants.

**List of Numeric Addresses and Data Lengths**

Ce paramètre fournit l'adresse numérique et la longueur des données devant être écrites. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**List of Data types**

Ce paramètre facultatif donne la description de type des valeurs dans le paramètre List of Data.

**List of Data**

Ce paramètre spécifie une ou plusieurs valeurs ou une ou plusieurs valeurs avec la révision d'objet à écrire. Les valeurs avec/sans révision d'objet sont concaténées dans l'ordre dans lequel elles sont décrites dans le paramètre précédent. Pour les listes de variables, il s'agit toujours de l'ordre de la définition de liste.

**Value**

Ce paramètre spécifie une valeur à écrire. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Value with Object Revision**

Ce paramètre spécifie une valeur à écrire plus la révision d'objet prévue de l'objet variable. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**List of Variable Data Access Statuses**

Ce paramètre spécifie un indicateur pour chaque variable à laquelle l'accès a eu lieu. Si un accès réussit, il convient que l'indicateur soit TRUE; autrement, il convient qu'il soit FALSE.

**List of Error Statuses**

Ce paramètre spécifie un statut d'erreur pour chaque variable individuelle ou variable dans une liste de variables pour laquelle l'accès a échoué. L'ordre des erreurs de statut est le même que l'ordre des objets identifiés dans la demande de service.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.4.4.5.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Write List est un service de "best effort". "Best effort" signifie que le service réussit si au moins une valeur est écrite.

Si l'utilisateur répondeur est capable d'écrire la valeur de l'une au moins des variables demandées, l'utilisateur retourne une primitive "response(+)" du service Write List.

Si l'utilisateur n'est pas capable d'écrire au moins l'une des valeurs, l'utilisateur produit une primitive "response (-)" du service Write List en indiquant la cause.

#### 6.2.4.4.6 Service Information report

##### 6.2.4.4.6.1 Vue d'ensemble du service

Ce service confirmé est utilisé par un processus application pour rapporter la valeur d'une variable au(x) destinataire(s) désigné(s) par l'AR.

##### 6.2.4.4.6.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 45.

**Tableau 45 – Service Information report**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Variable Specifier | C | C (=) |
| Key Attribute | S | S (=) |
| Key Attribute and Component ID | S | S (=) |
| Numeric Address and Data Length | S | S (=) |
| Value | M | M (=) |
| Object Revision | U | U (=) |
| Data type | U | U (=) |
| Timeliness | | C |
| Duplicate FAL PDU Body | | C |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Destination DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il donne l'adresse distante à laquelle il convient d'envoyer le rapport d'informations (Information Report) demandé.

**Source DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il identifie l'adresse source à partir de laquelle le rapport d'informations (Information Report) est à envoyer.

**Variable Specifier**

Ce paramètre de type sélecteur spécifie la variable, la liste de variables ou un élément d'une variable matrice ou enregistrement.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Component Identifier**

Ce paramètre identifie le champ d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur de champ d'une structure de données ou d'un indice dans une matrice. Ce paramètre peut identifier un composant avec un niveau d'imbrication supérieur à un, s'il est pris en charge par l'AP.

**Numeric Address and Data Length**

Ce paramètre fournit l'adresse numérique et la longueur de données rapportées.

**Value**

Ce paramètre spécifie la valeur. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Object Revision**

Ce paramètre facultatif spécifie la révision d'objet prévue de l'objet variable. Lorsqu'il est présent, il indique que l'utilisateur de service appelant souhaite que le serveur vérifie le paramètre Object Revision courant de la variable devant être mise à jour. S'il est pris en charge, le serveur ne met à jour la valeur d'une variable dans la liste que si son attribut Object Revision équivaut à la valeur de ce paramètre.

**Data type**

Ce paramètre facultatif spécifie le type de donnés des valeurs dans le paramètre Value.

**Timeliness**

Ce paramètre conditionnel indique le statut de cohérence temporelle de couche liaison de données pour l'objet "variable" ou "variable list" référencé. Il est présent si la cohérence temporelle est prise en charge sur l'AR spécifiée.

**Duplicate FAL PDU Body**

Ce paramètre conditionnel indique si, oui ou non, la réception d'une PDU de FAL dupliquée a été détectée par la couche liaison de données. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

#### 6.2.4.4.6.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

Un paramètre de cohérence temporelle est inclus dans la primitive "indication" si l'AR qui a acheminé le corps d'APDU prend en charge la cohérence temporelle.

### 6.2.4.4.7 Service Information report list

#### 6.2.4.4.7.1 Vue d'ensemble du service

Ce service non confirmé est utilisé par un processus application pour rapporter au(x) destinataire(s) désigné(s) par l'AR une liste de valeurs ou une liste de valeurs avec révision d'objet. Le nombre de variables référencées ou définies dans la liste relève d'une initiative locale de l'appareil.

#### 6.2.4.4.7.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 46.

**Tableau 46 – Service Information report list**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| List of Variable Specifiers | S | S (=) |
| Key Attribute | S | S (=) |
| Key Attribute and Component IDs | S | S (=) |
| List of Numeric Addresses and Data Lengths | S | S (=) |
| List of Data types | U | U (=) |
| List of Data | M | M (=) |
| Value | S | S (=) |
| Value with Object Revision | S | S (=) |
| Timeliness | | C |
| Duplicate FAL PDU Body | | C |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Destination DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il donne l'adresse distante à laquelle la liste de rapports d'informations (Information Report List) demandée est à envoyer.

**Source DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse source à partir de laquelle la liste de rapports d'informations (Information Report List) indiquée est à envoyer.

**List of Variable Specifiers**

Ce paramètre identifie individuellement plusieurs variables et/ou composants de variables devant être écrit(e)s ou une seule liste de variables devant être écrite. Le niveau d'imbrication des composants devant être écrits peut être supérieur ou égal à un.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Component ID(s)**

Ce paramètre identifie les composants d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur (ou des identificateurs) de chemin d'accès aux composants.

**List of Numeric Addresses and Data Lengths**

Ce paramètre fournit l'adresse numérique et la longueur de données des données devant être écrites. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**List of Data types**

Ce paramètre facultatif donne la description de type des valeurs dans le paramètre List of Data.

**List of Data**

Ce paramètre spécifie une ou plusieurs valeurs ou une ou plusieurs valeurs avec révision d'objet à écrire. Les valeurs avec/sans révision d'objet sont concaténées dans l'ordre dans lequel elles sont décrites dans le paramètre précédent. Pour les listes de variables, il s'agit toujours de l'ordre de la définition de liste.

**Value**

Ce paramètre spécifie une valeur à écrire. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Value with Object Revision**

Ce paramètre spécifie une valeur à écrire plus la révision d'objet prévue de l'objet variable. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Timeliness**

Ce paramètre conditionnel indique la cohérence temporelle de couche liaison de données pour la liste de valeurs. Il est présent si la cohérence temporelle est définie pour l'AR spécifiée.

**Duplicate FAL PDU Body**

Ce paramètre conditionnel indique si, oui ou non, la réception d'une PDU de FAL dupliquée a été détectée par la couche liaison de données. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

#### 6.2.4.4.7.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

Si l'objet spécifié devant être rapporté est un objet Variable List, les valeurs ou les valeurs avec révision d'objet des variables dans la liste de variables sont concaténées dans le paramètre List of Data dans l'ordre de leur apparition dans la liste de variables.

Un paramètre de cohérence temporelle est inclus dans la primitive "indication" si l'AR qui a acheminé le corps d'APDU prend en charge la cohérence temporelle.

### 6.2.4.4.8 Service Exchange

#### 6.2.4.4.8.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour écrire la valeur d'un objet Variable ou Variable List et lire la valeur d'un autre objet Variable ou Variable List en une seule et même opération. L'ordre de traitement de la lecture et de l'écriture par un utilisateur répondeur n'est pas spécifié par la FAL.

#### 6.2.4.4.8.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 47.

**Tableau 47 – Paramètres du service Exchange**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variable to Write | M | M (=) | | |
| Specifier (spécificateur) | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Element ID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Value | M | M (=) | | |
| Data type | U | U (=) | | |
| Object Revision | U | U (=) | | |
| Variable to Read | M | M (=) | | |
| Specifier (spécificateur) | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and ElementID | S | S (=) | | |
| Numeric Address and Data Length | S | S (=) | | |
| Data type Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Write Result | | | M | M (=) |
| Read Response | | | M | M (=) |
| Error Status | | | S | S (=) |
| Returned Value | | | S | S (=) |
| Value | | | M | M (=) |
| Data type | | | C | C (=) |
| Object Revision | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.4.3. | | | | |

### Argument

L'argument contient les paramètres de la demande du service.

### Specifier for Variable to Write

Ce paramètre spécifie la variable, la liste de variables ou le champ de variable devant être mis(e) à jour.

#### Key Attribute

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

#### Key Attribute and Element Identifier

Ce paramètre identifie le champ d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur de champ d'une structure de

données ou d'un indice dans une matrice. Ce paramètre peut identifier un élément avec un niveau d'imbrication supérieur à un, s'il est pris en charge par l'AP.

**Numeric Address/Data Length**

Ce paramètre fournit l'adresse numérique et la longueur de données des données devant être écrites. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**Value to Write**

Ce paramètre spécifie la valeur à écrire. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

NOTE Si une variable quelconque d'une liste de variables ne peut pas être mise à jour, aucune des variables dans l'objet Variable List ne sera mise à jour et l'écriture échouera.

**Data type**

Ce paramètre facultatif spécifie le type de donnés des valeurs à écrire dans le paramètre Value.

**Object Revision**

Ce paramètre facultatif spécifie la révision d'objet prévue de l'objet variable devant être mis à jour. Lorsqu'il est présent, il indique qu'il faut que l'AP contenant la variable compare la valeur de ce paramètre au paramètre Object Revision de la variable avant d'appliquer l'écriture.

**Variable Specifier**

Ce paramètre de type sélecteur spécifie la variable, la liste de variables ou le champ de variable devant être lu(e).

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Element Identifier**

Ce paramètre identifie le champ d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur de champ d'une structure de données ou d'un indice dans une matrice. Ce paramètre peut identifier un élément avec un niveau d'imbrication supérieur à un, s'il est pris en charge par l'AP.

**Numeric Address/Data Length**

Ce paramètre fournit l'adresse numérique et la longueur de données des données devant être lues. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**Data type Requested**

Ce paramètre facultatif indique qu'il convient que le type de données de chaque variable soit retourné avec les données.

**Object Revision Requested**

Ce paramètre facultatif est utilisé pour demander que la valeur de l'attribut "Object Revision" (révision d'objet) soit retournée. Une valeur TRUE indique que la révision d'objet est souhaitée. Une valeur FALSE indique que la révision d'objet n'est pas souhaitée. Si Object Revision est demandé, mais n'est pas pris en charge pour l'objet spécifié, le service échoue.

**Result (+)**

Le paramètre Result (+) indique que la demande de service a réussi.

**Write Status**

Ce paramètre indique si, oui ou non, la variable a été mise à jour comme cela était demandé. Si elle n'a pas été mise à jour, il indique la cause la plus probable de l'échec de l'opération en utilisant la classe d'erreurs et le code d'erreur définis pour le service d'écriture.

**Read Response**

Ce paramètre fournit la valeur de données correspondant au paramètre Variable Specifier to Read dans la primitive "request". Ce paramètre indique le résultat de l'opération de lecture.

**Error Status**

Ce paramètre est utilisé pour indiquer que l'accès à l'objet spécifié n'a pas réussi. Il indique la cause la plus probable de l'échec de l'opération en utilisant la classe d'erreurs et le code d'erreur définis pour le service de lecture.

**Returned Value**

Ce paramètre spécifie la valeur de l'objet variable ou de l'objet liste de variables référencé. Si la variable identifiée était un objet Variable List, ce paramètre contiendra un jeu concaténé de valeurs pour chaque variable de la liste.

**Value**

Ce paramètre spécifie la valeur lue. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

NOTE Comme il n'y aucun identificateur dans une liste concaténée de valeurs pour un objet Variable List, un échec de lecture pour toute variable de l'objet Variable List entraînera l'échec du service.

**Data type**

Ce paramètre facultatif indique que le data type de chaque valeur est retourné.

**Object Revision**

Ce paramètre conditionnel spécifie la révision d'objet relative à l'objet spécifié. Il est présent si la révision d'objet a été demandée.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.4.4.8.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Exchange est un service de "best effort". "best effort" signifie que le service réussit si les opérations de lecture ou d'écriture demandées réussissent.

Si une liste de variables a été spécifiée pour la variable à écrire, la valeur et la révision d'objet (lorsqu'elles sont présentes) de chaque variable dans la liste de variables sont concaténées dans le corps d'APDU de demande Exchange dans l'ordre de l'apparition des variables dans la liste de variables.

Si une liste de variables a été spécifiée pour la variable à lire, la valeur et la révision d'objet (lorsqu'elles sont présentes) de chaque variable dans la liste de variables sont concaténées dans le corps d'APDU de réponse Exchange dans l'ordre de l'apparition des variables dans la liste de variables.

#### 6.2.4.4.9 Service Exchange list

##### 6.2.4.4.9.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour écrire plusieurs objets "variable" ou composants individuels de plusieurs objets "variable" ou un seul objet "variable list" et pour lire plusieurs objets "variable" ou composants individuels de plusieurs objets "variable" ou un seul objet " variable list" en une seule et même opération. L'ordre de traitement de la lecture et de l'écriture par un utilisateur répondeur n'est pas spécifié par la FAL.

##### 6.2.4.4.9.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 48.

**Tableau 48 – Paramètres du service Exchange list**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Variables to Write | M | M (=) | | |
| List of Variable Specifiers | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component IDs | S | S (=) | | |
| List of Numeric Addresses and Data Lengths | S | S (=) | | |
| List of Data types | U | U (=) | | |
| List of Data | M | M (=) | | |
| Value | S | S (=) | | |
| Value with Object Revision | S | S (=) | | |
| Variables to Read | M | M (=) | | |
| List of Variable Specifiers | S | S (=) | | |
| Key Attribute | S | S (=) | | |
| Key Attribute and Component IDs | S | S (=) | | |
| List of Numeric Addresses and Data Lengths | S | S (=) | | |
| Data type Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List of Write Access Results | | | M | M (=) |
| List of Error Statuses | | | C | C (=) |
| List of Read Access Results | | | M | M (=) |
| List of Data types | | | C | C (=) |
| List of Data | | | M | M (=) |
| Error Status | | | S | S (=) |
| Value | | | S | S (=) |
| Value With Object Revision | | | S | S (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE Voir la Note en 3.8.4.3.

**Argument**

L'argument contient les paramètres de la demande du service.

**Variables to Write**

Ce paramètre spécifie les informations relatives aux variables devant être mises à jour.

**List of Variable Specifiers**

Ce paramètre identifie individuellement plusieurs variables et/ou composants de variables devant être écrit(e)s ou une seule liste de variables devant être écrite. Le niveau d'imbrication des composants devant être écrits peut être supérieur ou égal à un.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Component ID(s)**

Ce paramètre identifie le composant d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur (ou des identificateurs) de chemin d'accès aux composants.

**List of Numeric Addresses and Data Lengths**

Ce paramètre fournit l'adresse numérique et la longueur de données des données devant être écrites. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**List of Data types**

Ce paramètre facultatif donne la description de type des valeurs dans le paramètre List of Data.

**List of Data**

Ce paramètre spécifie une ou plusieurs valeurs ou une ou plusieurs valeurs avec révision d'objet à écrire. Les valeurs avec/sans révision d'objet sont concaténées dans l'ordre dans lequel elles sont décrites dans le paramètre précédent. Pour les listes de variables, il s'agit toujours de l'ordre de la définition de liste.

**Value**

Ce paramètre spécifie une valeur à écrire. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Value with Object Revision**

Ce paramètre spécifie une valeur à écrire plus la révision d'objet prévue de l'objet variable. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Variables to Read**

Ce paramètre contient les informations relatives aux variables devant être lues.

**List of Variable Specifiers**

Ce paramètre identifie individuellement plusieurs variables et/ou composants de variables devant être écrit(e)s ou une seule liste de variables devant être lue. Le niveau d'imbrication des composants devant être lus peut être supérieur ou égal à un.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Key Attribute and Component ID(s)**

Ce paramètre identifie le composant d'une variable de data type construit par une combinaison de l'un de ses attributs-clés et de l'identificateur (ou des identificateurs) de chemin d'accès aux composants.

**List of Numeric Addresses and Data Lengths**

Ce paramètre fournit l'adresse numérique et la longueur de données des données devant être lues. Le mapping entre ce paramètre et l'adresse mémoire réelle dans un système réel relève d'une initiative locale.

**Data type Requested**

Ce paramètre facultatif est utilisé pour demander que le data type soit retourné. Une valeur TRUE indique que le type de données est souhaité. Une valeur FALSE indique que le type de donnée n'est pas souhaité.

**Object Revision Requested**

Ce paramètre facultatif est utilisé pour demander que la valeur de l'attribut "Object Revision" (révision d'objet) soit retournée. Une valeur TRUE indique que la révision d'objet est souhaitée. Une valeur FALSE indique que la révision d'objet n'est pas souhaitée.

**Result (+)**

Le paramètre Result (+) indique que la demande de service a réussi.

**List of Write Access Results**

Ce paramètre spécifie un indicateur pour chaque variable à laquelle l'accès a eu lieu. Si un accès réussit, il convient que l'indicateur soit TRUE; autrement, il convient qu'il soit FALSE.

**List of Error Statuses**

Ce paramètre spécifie un statut d'erreur pour chaque variable individuelle ou variable dans une liste de variables pour laquelle l'accès a échoué. L'ordre des erreurs de statut est le même que l'ordre des objets identifiés dans la demande de service.

**List of Read Access Results**

Ce paramètre spécifie un indicateur pour chaque variable à laquelle l'accès a eu lieu. Si un accès réussit, il convient que l'indicateur soit TRUE; autrement, il convient qu'il soit FALSE.

**List of Data types**

Ce paramètre facultatif donne la description de type des valeurs dans le paramètre List of Data.

**List of Data**

Ce paramètre spécifie un(e) ou plusieurs statuts d'erreur, valeurs ou valeurs avec révision d'objet lu(e)s. Les statuts d'erreur ou valeurs avec/sans révision d'objet sont concaténé(e)s dans l'ordre dans lequel ils/elles ont été décrit(e)s dans la demande. Pour les listes de variables, il s'agit toujours de l'ordre de la définition de liste. Pour chaque variable ou une variable dans une liste de variables à laquelle l'accès a réussi, il convient que la donnée retournée soit la valeur ou la valeur avec révision d'objet de l'objet variable (si elle est demandée). Autrement, il convient qu'elle soit un statut d'erreur. Si la révision d'objet est demandée et l'objet variable atteint ne prend pas en charge l'attribut Object Revision, il convient que la donnée retournée soit également un statut d'erreur.

**Error Status**

Ce paramètre indique la cause la plus probable de l'échec de l'opération en utilisant la classe d'erreurs et le code d'erreur définis pour le service de lecture.

**Value**

Ce paramètre spécifie une valeur lue. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune

des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Value with Object Revision**

Ce paramètre spécifie une valeur lue plus la révision d'objet de l'objet variable. Pour les variables, ce paramètre spécifie la valeur de la variable. Pour les listes de variables, ce paramètre spécifie les valeurs de chacune des variables de la liste concaténées ensemble dans l'ordre de leur apparition dans la liste.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.4.4.9.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Exchange est un service "best effort". "Best effort" signifie que le service réussit si les opérations de lecture ou d'écriture demandées réussissent.

L'opération d'écriture réussit si l'accès à n'importe quel de variables de la liste ou n'importe quelle variable dans la liste de variables réussit.

L'opération de lecture réussit si n'importe quelle variable atteinte dans la liste ou n'importe quel de variables de la liste de variables réussit.

### 6.2.5 ASE Event

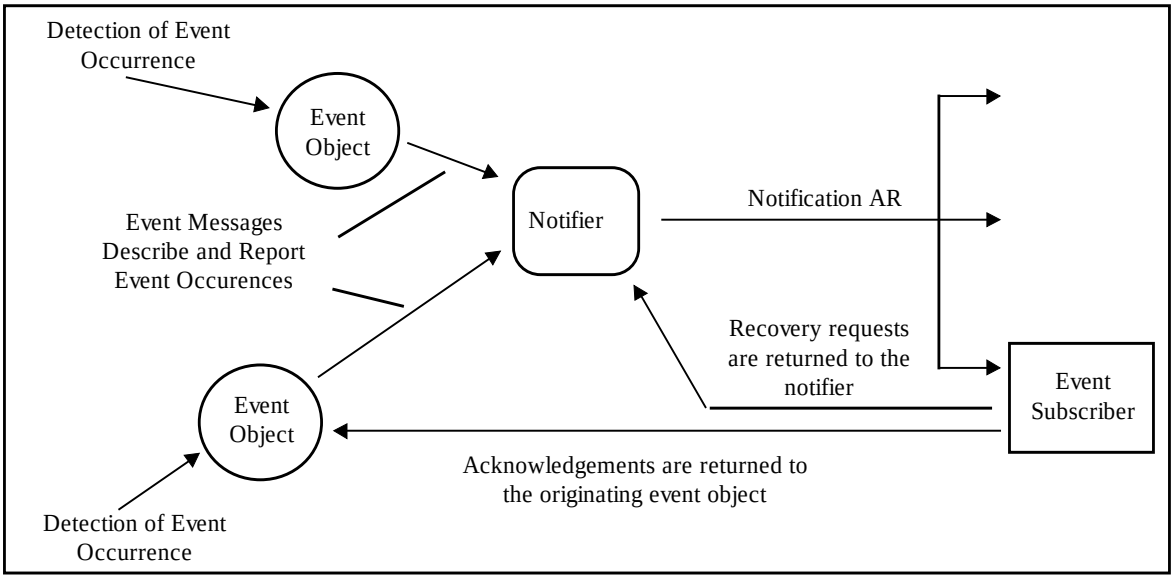
#### 6.2.5.1 Vue d'ensemble

Le modèle d'événement de la FAL définit trois objets liés à un événement, à savoir les objets Event (événement) pour la définition des événements et de leurs messages, les notificateurs d'événements pour la distribution des messages d'événement et les listes d'événements pour l'acquisition d'informations récapitulatives relatives à des événements glanées à partir d'un groupe d'objets Event.

Les objets Event sont utilisés pour définir des messages utilisés pour rapporter des apparitions d'événements Les messages Event contiennent des informations qui identifient et décrivent les apparitions d'événements. Pour simplifier le traitement des messages d'événements, les objets Event sont définis pour générer des messages utilisant l'un de quatre formats définis.

NOTE Par exemple, il peut être défini des objets Event qui génèrent des messages d'événements contenant un indicateur de temps et/ou un compteur des messages pour les apparitions d'événements.

Les notificateurs sont chargés de rassembler les messages d'événements provenant d'objets Event et d'en distribuer un ou plusieurs dans une seule invocation du service de notification d'événement de la FAL. Le nombre des messages d'événement qui peuvent être présentés dans une seule invocation de service est limité par la taille maximale des APDU qui peut être transférée par l'AR. La Figure 4 illustre ce concept.




**Légende**

| Anglais | Français |
|---|---|
| Detection of Event Occurrence | Détection de l'apparition d'un événement |
| Event Object | Objet Event (événement) |
| Event Messages Describe and Report Event Occurences | Décrire les messages d'événement et rapporter les apparitions d'événement |
| Notifier | Notificateur |
| Notification AR | AR de Notification |
| Recovery requests are returned to the notifier | Les notifications de récupération sont retournées au notificateur |
| Event Subscriber | Abonné de l'événement |
| Acknowledgements are returned to the originating event object | Les acquittements sont retournés à l'objet événement émetteur |
| Detection of Event Occurrence | Détection d'apparition d'événement |

**Figure 4 – Vue d'ensemble du modèle d'événement**

Afin de fournir de la flexibilité dans le processus de notification d'événement, les notificateurs peuvent être définis comme utilisant l'un de quatre types de notifications d'événements pour transporter les messages d'événements. Une fois défini, un notificateur utilise toujours le même type de notification d'événement pour transporter un ou plusieurs types différents de messages d'événements.

Si un processus application échoue à recevoir une ou plusieurs notifications d'événement, une récupération de notification est fournie pour qu'il demande la réémission par le notificateur.

En outre, le service Get Event Summary est fourni pour permettre à un processus application d'interroger un ou plusieurs objets Event pour sélectionner ceux qui satisfont à certains critères relatifs à l'état courant des objets Event. Un objet Event list (liste d'événements) est défini pour permettre que des objets Event soient regroupés ensemble afin de simplifier le processus d'interrogation.

La classe AR Endpoint (c'est-à-dire la classe des relations des points d'extrémité entre applications) utilisée pour distribuer des notifications d'événement est spécifiée par l'un des

attributs de la classe Notifier (notificateur) Un attribut d'événement est utilisé pour spécifier l'AR qui distribue les notifications.

Les AR mises en file d'attente sont utilisées le plus souvent, car elles empêchent que des notifications d'événement soient écrasées en écriture avant d'être lues. Lorsque la notification a pour but de signaler un changement d'état de l'AP de l'appareil de terrain, il est permis d'utiliser des AR placées en buffer.

NOTE Lorsqu'aucun événement ne s'est produit pendant une durée spécifiée, les notificateurs peuvent choisir de générer une notification d'événement "heartbeat" spéciale qui indique aux abonnés que le notificateur est toujours opérationnel. Ces notifications spéciales ressemblent à la dernière notification normale envoyée, avec l'exception qu'elles ne contiennent pas les messages d'événements. Les notifications heartbeat sont définies pour permettre à des abonnes à des événements de déterminer s'ils ont manqué une notification d'événement.

Dans ce modèle, les processus application sont chargés de fournir les fonctions pour les objets Event, Notifier et Event List (liste d'événements) tandis que la FAL est chargée de fournir les services de communication conçus spécifiquement pour eux. Le processus application détecte les événements, construit des messages d'événement et les agrège ensemble. Il distribue l'ensemble agrégé en utilisant le service de notification d'événement de la FAL. En l'extrémité de réception, les processus application de l'abonné peuvent utiliser l'un de deux services d'acquittement d'événement pour acquitter les apparitions d'événement.

Le reste du présent paragraphe spécifie les définitions de classes et les services pour les objets Event, Notifier et Event List.

#### 6.2.5.2 Spécification de la classe modèle Event

##### 6.2.5.2.1 Spécification de la classe Event

###### 6.2.5.2.1.1 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE EVENT** |
| **CLASS:** | EVENT | | |
| **CLASS ID:** | | | 4 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut: | Event Status |
| 1.1 | (o) | Attribut: | Active (TRUE,FALSE) |
| 1.2 | (o) | Attribut: | Detected (TRUE,FALSE) |
| 1.3 | (m) | Attribut: | Enabled (TRUE,FALSE) |
| 1.4 | (o) | Attribut: | Acknowledged (TRUE,FALSE) |
| 2 | (m) | Attribut: | Message Type (SIMPLE,<br>REPORTED EVENT COUNT,<br>EVENT DETECTION TIME,<br>COMPOSITE,<br>SIMPLE WITH DATA,<br>REPORTED EVENT COUNT WITH DATA,<br>EVENT DETECTION TIME WITH DATA,<br>COMPOSITE WITH DATA) |
| 3 | (m) | Attribut: | Access Privilege |
| 3.1 | (m) | Attribut: | Password |
| 3.2 | (m) | Attribut: | Access Groups |
| 3.3 | (m) | Attribut: | Access Rights |
| 4 | (o) | Attribut: | Last Reported Event Info |
| 4.1 | (o) | Attribut: | Count |
| 4.2 | (o) | Attribut: | Detection Time |
| 5 | (c) | Contrainte: | Message Type = SIMPLE WITH DATA \|<br>REPORTED EVENT COUNT WITH DATA \| |

| | | | |
|---|---|---|---|
| | | | EVENT DETECTION TIME WITH DATA \| COMPOSITE WITH DATA) |
| 5.1 | (c) | Attribut: | Event Data Specifier |
| 5.1.1 | (s) | Attribut: | Variable ID |
| 5.1.2 | (s) | Attribut: | Data type ID and Length |
| 6 | (c) | Contrainte: | Event Status.Acknowledgment Supported = TRUE |
| 6.1 | (o) | Attribut: | Acknowledgment Data Specifier |
| 6.1.1 | (s) | Attribut: | Variable ID |
| 6.1.2 | (s) | Attribut: | Data type ID and Length |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Acknowledge Event |
| 2 | (o) | OpsService: | Acknowledge Event List |
| 3 | (o) | OpsService: | Enable Event |
| 4 | (o) | OpsService: | Enable Event List |
| 5 | (o) | OpsService: | Get Event Summary |
| 6 | (o) | OpsService: | Get Event Summary List |
| 7 | (o) | OpsService: | Query Event Summary List |

#### 6.2.5.2.1.2 Attributs

**Event Status**

Cet attribut indique le statut de l'événement. Le statut est indiqué par une combinaison d'attributs Boolean (booléens): active, detected, enabled, acknowledgment support et acknowledged.

**Active**

Cet attribut facultatif indique, lorsqu'il est TRUE, que les événements peuvent être détectés. S'il est FALSE, les apparitions d'événement ne sont pas détectées.

NOTE 1 L'objet Notifier possède aussi un attribut enable («permettre») utilisé pour commander si, oui ou non, les notifications d'événement sont à générer.

**Detected**

Cet attribut facultatif indique, lorsqu'il est TRUE, que les conditions de détection d'une apparition d'événement sont vraies (TRUE).

**Enabled**

Cet attribut indique, lorsqu'il est TRUE, que les messages d'événement sont à rapporter au notificateur concerné à la suite de la détection d'apparitions d'événements. S'il est FALSE, il ne faut pas générer de message d'événement ni les rapporter lorsque des apparitions d'événement sont détectées.

NOTE 2 L'objet Notifier possède aussi un attribut enable («autoriser») utilisé pour commander si, oui ou non, les notifications d'événement sont à générer.

**Acknowledged**

Cet attribut facultatif indique, lorsqu'il est TRUE, que tous les messages d'événement générés pour cet objet Event ont été acquittés. Cet attribut sera également TRUE si aucun message d'événement n'a été généré ou si l'objet Event ne prend pas en charge l'acquittement. La valeur FALSE indique que l'objet Event attend un acquittement.

**Message Type**

Cet attribut spécifie le type de message d'événement qui est généré lorsqu'un événement se produit. Les types de messages définis et leur structure sont:

| Type | Contenu |
|---|---|
| SIMPLE MSG | Event Numeric ID |

| | |
|---|---|
| REPORTED EVENT COUNT | Event Numeric ID |
| | Last Detected Event Info.Count |
| EVENT DETECTION TIME | Event Numeric ID |
| | Last Reported Event Info.Detection Time |
| COMPOSITE | Event Numeric ID |
| | Last Reported Event Info.Count |
| | Last Reported Event Info.Detection Time |
| SIMPLE MSG WITH DATA | Event Numeric ID |
| | Event Data Identified by the Event Data Specifier |
| REPORTED EVENT COUNT WITH DATA | Event Numeric ID |
| | Last Detected Event Info.Count |
| | Event Data Identified by the Event Data Specifier |
| EVENT DETECTION TIME WITH DATA | Event Numeric ID |
| | Last Reported Event Info.Detection Time |
| | Event Data Identified by the Event Data Specifier |
| COMPOSITE WITH DATA | Event Numeric ID |
| | Last Reported Event Info.Count |
| | Last Reported Event Info.Detection Time |
| | Event Data Identified by the Event Data Specifier |

**Access Privilege**

Cet attribut spécifie les contrôles d'accès définis pour cet événement. Il se compose des éléments suivants:

**Password**

Cet attribut spécifie le mot de passe pour les droits d'accès. Sa valeur est null s'il n'est pas utilisé.

**Access Groups**

Cet attribut identifie lesquels des huit groupes d'accès définis par un utilisateur sont définis pour l'événement. Plus d'un groupe d'accès peut être défini.

**Access Rights**

Cet attribut définit le type d'accès défini pour l'événement. Les valeurs valides sont:

- Droit d'acquitter le mot de passe enregistré
- Droit d'activer/désactiver le mot de passe enregistré
- Droit d'acquitter les groupes d'accès
- Droit d'activer/désactiver les groupes d'accès
- Droit d'acquitter tous les partenaires de communication
- Droit d'activer/désactiver tous les partenaires de communication

**Last Reported Event Info**

Cet attribut facultatif spécifie le compte et l'instant de détection du dernier événement rapporté.

#### Count

Cet attribut facultatif compte le nombre d'apparitions d'événement rapportées. Il est incrémenté de un pour chaque apparition d'événement détectée et rapportée. Lorsque l'objet Event est disabled (désactivée), les apparitions d'événement détectées ne sont pas comptées.

Lorsque la valeur de cet attribut dépasse son maximum, elle passe à 1, sautant 0. La valeur 0 est toujours utilisée pour indiquer qu'aucun message d'événement n'a été généré. Si l'attribut Message Type («type de message») est mis à REPORTED EVENT COUNT ou COMPOSITE, sa valeur mise à jour est incluse dans les messages d'événements.

NOTE 3 La présence de cet attribut est indépendante de la question de savoir si, oui ou non, il contient un message d'événement. Sa présence indique que cet attribut est accessible en utilisant les services de gestion de la FAL. La FAL n'est pas impliquée à l'incrémentation de la valeur de cet attribut. Par conséquent, la FAL ne sait pas à quel moment sa valeur dépasse son maximum ou si elle le dépasse correctement.

#### Detection Time

Cet attribut facultatif spécifie l'instant auquel le dernier événement rapporté a été détecté. Si l'attribut Message Type («type de message») est mis à EVENT DETECTION TIME ou COMPOSITE, sa valeur est incluse dans les messages d'événements.

NOTE 4 La présence de cet attribut est indépendante de la question de savoir si, oui ou non, il contient un message d'événement. Sa présence indique que cet attribut est accessible en utilisant les services de gestion de la FAL.

### Event Data Specifier

Cet attribut de type conditionnel spécifie les données d'utilisateur devant être incluses dans le message d'événement. Les données d'événement sont spécifiées soit en identifiant un objet variable devant être incorporé dans un message d'événement, soit en spécifiant simplement leur data type et leur longueur. Cet attribut est présent lorsque le Message Type indique que les données d'événement sont à inclure dans les messages envoyés.

#### Variable ID

Cet attribut de type sélection identifie un objet Variable ou un objet Variable List contenu dans le même AP que l'événement dont la valeur est à incorporer dans un message d'événement.

#### Data type ID and Length

Cet attribut de type sélection identifie le data type et la longueur des données d'utilisateur qui sont à incorporer dans un message d'événement. En fonction du data type, la longueur peut être définie par le data type.

### Acknowledgment Data Specifier

Cet attribut facultatif spécifie les données d'utilisateur devant être incluses dans le message d'acquittement. Les données d'acquittement sont spécifiées soit en identifiant un objet variable devant être incorporé dans un message d'acquittement, soit en spécifiant simplement leur data type.

#### Variable ID

Cet attribut de type sélection identifie un objet Variable ou un objet Variable List contenu dans le même AP que l'événement dont la valeur est à incorporer dans un message d'acquittement.

**Data type ID and Length**

Cet attribut de type sélection identifie le data type et la longueur des données d'utilisateur qui sont à incorporer dans un message d'acquittement.

#### 6.2.5.2.1.3 Services

Tous les services définis pour cette classe sont facultatifs. Lorsqu'une instance de la classe est définie, l'un au moins de ces services est à sélectionner.

**Acknowledge Event**

Ce service confirmé facultatif est fourni pour permettre à un abonné d'événement d'accuser réception d'un message d'événement.

**Acknowledge Event List**

Ce service confirmé facultatif est fourni pour permettre à un abonné d'accuser réception d'une liste de messages d'événement générés par un ou plusieurs objets Event spécifiés.

**Enable Event**

Ce service facultatif est fourni pour un permettre à un abonné d'un événement d'activer un événement.

**Enable Event List**

Ce service facultatif est fourni pour un permettre à un abonné d'un événement d'activer une liste d'événements.

**Get Event Summary**

Ce service facultatif fournit la capacité de récupérer des informations récapitulatives pour un seul événement.

**Get Event Summary List**

Ce service facultatif fournit la capacité de récupérer des informations récapitulatives pour une liste d'événements.

**Query Event Summary List**

Ce service facultatif fournit la capacité d'interroger un ensemble d'événements pour des informations récapitulatives selon des critères de sélection.

### 6.2.5.2.2 Spécification de la classe "Event List" (liste d'événements)

#### 6.2.5.2.2.1 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE EVENT** |
| **CLASS:** | EVENT LIST | | |
| **CLASS ID:** | | | 17 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut: | Number of Entries |
| 2 | (m) | Attribut: | List Of Events |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Get Event Summary List |
| 2 | (o) | OpsService: | Query Event Summary List |

#### 6.2.5.2.2.2 Attributs

**Number of Entries**

Cet attribut spécifie le nombre d'événements dans la liste.

**List of Events**

Cet attribut identifie les objets Event qui sont contenus dans la liste d'événement.

#### 6.2.5.2.2.3 Services

Tous les services définis pour cette classe sont facultatifs. Lorsqu'une instance de la classe est définie, l'un au moins de ces services est à sélectionner.

**Get Event Summary List**

Ce service facultatif fournit la capacité de récupérer des informations récapitulatives pour une liste d'événements.

**Query Event Summary**

Ce service facultatif fournit la capacité d'interroger un ensemble d'événements pour des informations récapitulatives selon des critères de sélection.

### 6.2.5.2.3 Spécification de classe "Notifier"

#### 6.2.5.2.3.1 Modèle formel

Le présent paragraphe fournit les définitions de classe formelle pour les objets "Notifier". Les notificateurs fournissent la capacité d'envoyer des notifications d'événement composées de messages d'apparitions d'événement issus d'un ou plusieurs objets Event.

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE EVENT** |
| **CLASS:** | NOTIFIER | | |
| **CLASS ID:** | | | 18 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut: | Enabled (TRUE,FALSE) |
| 2 | (m) | Attribut: | Notification AREP Class |
| 3 | (m) | Attribut: | Notification AREP |
| 4 | (m) | Attribut: | Notification Type (BASIC, SEQUENCED, TIME OF NOTIFICATION, COMPOUND) |
| 5 | (o) | Attribut: | Contained Message Type (SIMPLE, REPORTED EVENT COUNT, EVENT DETECTION TIME, COMPOSITE) |
| 6 | (o) | Attribut: | Contained Event Data (TRUE, FALSE) |
| 7 | (o) | Attribut: | Last Notification Sequence Number |
| 8 | (o) | Attribut: | List Of Events |
| **SERVICES:** | | | |
| 1 | (m) | OpsService: | Notification d'événement |
| 2 | (o) | OpsService: | Notification Recovery |

#### 6.2.5.2.3.2 Attribut

**Enabled**

Lorsqu'il est TRUE, cet attribut indique que l'objet "Notifier" est activé. Lorsqu'il est "enabled" (activé), l'objet Notifier regroupe les messages d'événement provenant d'un ou plusieurs objets Event et les présente dans une seule invocation du service de notification d'événement à la FAL pour leur acheminement.

Lorsqu'il est FALSE, l'objet Notifier est désactivé. Lorsqu'il est désactivé, le notificateur est inactif. Il ne produit aucune demande de service de notification d'événement.

**Notification AREP Class**

Cet attribut identifie la classe de l'AREP requis pour acheminer des notifications d'événements.

**Notification AREP**

Cet attribut identifie l'AREP configuré pour acheminer des notifications d'événements. Cet AREP est également l'AREP utilisé pour rapporter les notifications d'événements générées comme résultat d'une demande de récupération d'événement.

**Notification Type**

Cet attribut spécifie le type de notification émis par le notificateur. Les types de notification définis sont:

**Type Contenu**
BASIC Notifier ID
SEQUENCED Notifier ID
Sequence Number
TIME OF NOTIFICATION Notifier ID
Notification Time-Tag
COMPOUND Notifier ID
Sequence Number
Time-Tag

**Contained Message Type**

Cet attribut spécifie les types de messages d'événement qui peuvent être contenus dans des notifications produites par le notificateur. Les types de message d'événement sont définis par l'objet Event. Lorsqu'une notification d'événement est construite par cet objet Notifier, les types de messages d'événement qui peuvent être contenus dans la notification d'événement sont définis par cet attribut. Les types définis sont:

| **Description de Contained Message Type** | |
|---|---|
| SIMPLE | Tous les messages d'événement dans toutes les notifications sont de type SIMPLE. |
| REPORTED EVENT COUNT | Tous les messages d'événement dans toutes les notifications sont du type REPORTED EVENT COUNT. |
| EVENT DETECTION TIME | Tous les messages d'événement dans toutes les notifications sont du type EVENT DETECTION TIME. |
| COMPOSITE | Tous les messages d'événement dans toutes les notifications sont du type COMPOSITE. |
| SIMPLE WITH DATA | Tous les messages d'événement dans toutes les notifications sont du type SIMPLE. |
| REPORTED EVENT COUNT WITH DATA | Tous les messages d'événement dans toutes les notifications sont du type REPORTED EVENT COUNT WITH DATA. |
| EVENT DETECTION TIME WITH DATA | Tous les messages d'événement dans toutes les notifications sont du type EVENT DETECTION TIME WITH DATA. |

| | |
|---|---|
| COMPOSITE WITH DATA | Tous les messages d'événement dans toutes les notifications sont du type COMPOSITE WITH DATA. |

**Contained Event Data**

Cet attribut, lorsqu'il est TRUE, indique que les messages d'événement acheminés par le notificateur peuvent contenir des données d'événement. Les messages d'événement pour chacun des types de message d'événement peuvent contenir des données. Cet attribut indique si, oui ou non, les messages d'événement qui sont définis pour un objet Event pour contenir des données sont capables d'être transférés dans une notification d'événement générée par cet objet Notifier.

**Last Notification Sequence Number**

L'attribut conditionnel spécifie le dernier numéro de séquence utilisé. Il est incrémenté pour chaque invocation du service de notification d'événement. Lorsque la valeur de cet attribut dépasse son maximum, elle passe à 1, sautant 0. La valeur 0 est toujours utilisée pour indiquer qu'aucune notification d'événement n'a été générée. Il est obligatoire lorsque la valeur du type de notification est SEQUENCE ou COMPOUND.

NOTE La FAL n'effectue pas de vérification pour voir si le dépassement de maximum a été accompli correctement.

**List of Events**

Cet attribut facultatif identifie les événements qui sont configurés pour ce notificateur.

#### 6.2.5.2.3.3 Services

**Event Notification**

Ce service est fourni pour permettre à un objet Notifier de rapporter l'apparition d'un ou plusieurs événements dans une seule invocation de service.

**Notification Recovery**

Ce service facultatif fourni pour permettre à un ou plusieurs abonnés à des notifications d'événement de demander que le notificateur renvoie une ou plusieurs notifications d'événements qu'il a retenues. Si ce service est pris en charge, l'attribut Sequence Number est requis.

### 6.2.5.3 Spécification de services pour l'ASE Event

#### 6.2.5.3.1 Services pris en charge

Ce paragraphe contient la définition de services qui sont propres à cet ASE. Les services définis pour cet ASE sont:

Acknowledge Event

Acknowledge Event List

Enable Event

Enable Event List

Event Notification

Notification Recovery

Get Event Summary

Get Event Summary List

Query Event Summary

#### 6.2.5.3.2 Acknowledge event

##### 6.2.5.3.2.1 Vue d'ensemble du service

Ce service permet l'acquittement d'un événement rapporté.

##### 6.2.5.3.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 49.

**Tableau 49 – Acknowledge event**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Reported Event Count | M | M (=) | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande de service.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Reported Event Count**

Ce paramètre est utilisé pour sélectionner un message d'événement généré précédemment par le compte d'événements rapportés.

**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.

##### 6.2.5.3.2.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

L'association entre l'événement identifié dans ce service et l'objet Event correspondant dans l'AP de réception est accomplie par l'utilisateur de la FAL.

#### 6.2.5.3.3 Service Acknowledge event list

##### 6.2.5.3.3.1 Vue d'ensemble du service

Ce service confirmé est utilisé par un AP recevant un message d'événement pour accuser réception d'un ou plusieurs messages d'événement. Le service contient un paramètre Acknowledgment Policy (politique d'acquittement) qui indique pour chaque message d'événement spécifié si seul le message d'événement en question est à acquitter ou si tous les messages d'événement qui ont été générés par le même objet Event jusques et y compris l'événement spécifié sont à acquitter. Le service Acknowledge Event List est un service de "best effort".

##### 6.2.5.3.3.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 50.

**Tableau 50 – Paramètres du service Acknowledge event list**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Acknowledgment Policy | C | C (=) | | |
| Messages to Ack | M | M (=) | | |
| List Of Event ID | M | M (=) | | |
| List of Message Specifier | U | U (=) | | |
| Reported Event Count | U | U (=) | | |
| Event Detection Time | U | U (=) | | |
| List Of Acknowledgment Data | U | U (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List Of Access Result | | | M | M (=) |
| List Of Error | | | C | C (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Acknowledgment Policy**

Ce paramètre conditionnel indique si seul le message spécifié est à acquitter ou si tous les messages jusqu'au message spécifié compris sont à acquitter. Il n'est présent que lorsque le paramètre List of Messages to Ack identifie un seul message.

**Messages to Ack**

Ce paramètre spécifie les messages de tous les événements devant être acquittés.

**List of Event ID**

Ce paramètre identifie les événements acquittés par ce service.

**List of Message Specifier**

Ce paramètre facultatif spécifie les messages qui sont à acquitter pour chaque événement par cette invocation de service. Si ce paramètre n'est pas présent, tous les messages pour l'événement sont à acquitter.

Si la politique d'acquittement indique ONLY THE SPECIFIED, seul le message identifié par ce paramètre est à acquitter.

Si la politique d'acquittement indique UP TO AND INCLUDING, le message identifié par ce paramètre et tous les messages générés avant lui sont à acquitter.

**Reported Event Count**

Ce paramètre facultatif est utilisé pour sélectionner un message d'événement généré précédemment par le compte d'événements rapportés.

**Event Detection Time**

Ce paramètre facultatif est utilisé pour sélectionner un message d'événement généré précédemment au moment de la détection d'événement.

**List of Acknowledgment Data**

Ce paramètre facultatif spécifie les données utilisateur devant être incluses dans un acquittement d'événement en plus de celles utilisées pour identifier l'apparition d'événement. Le type de ces données est spécifié par l'attribut Acknowledgement Data Specifier de l'objet Event.

**Result (+)**

Cette sélection informe le client que le serveur était capable d'acquitter avec succès au moins un objet Event parmi ceux figurant dans la demande.

**List of Access Result**

Ce paramètre spécifie le résultat échec/succès de tous les objets Event acquittés.

**List of Error**

Ce paramètre indique la cause de l'échec pour chaque acquittement ayant échoué.

**Result(–)**

Cette sélection indique que la demande de service a échoué.

#### 6.2.5.3.3.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

L'association entre les événements référencés par le service Event Acknowledgement et l'objet Event correspondant dans l'AP de réception est accomplie par l'utilisateur de la FAL.

### 6.2.5.3.4 Service Enable event

#### 6.2.5.3.4.1 Vue d'ensemble du service

Ce service permet l'activation ou la désactivation de l'objet Event.

#### 6.2.5.3.4.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 51.

**Tableau 51 – Enable event**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Enable Flag | M | M (=) | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres spécifiques au service relatifs à la demande du service.

**Key Attribute**

Ce paramètre identifie la variable ou liste de variables par l'un de ses attributs-clés.

**Enable Flag**

Ce paramètre Boolean, lorsqu'il est TRUE, indique que l'objet Event est à activer, et lorsqu'il est FALSE, que l'objet Event est à désactiver.

**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.

#### 6.2.5.3.4.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.5.3.5 Service Event notification

#### 6.2.5.3.5.1 Vue d'ensemble du service

Ce service non confirmé est utilisé par un notificateur d'un AP de FAL à notifier à d'autres AP qu'un ou plusieurs événements se sont produits. Lorsque ce service est utilisé sans inclure une liste de messages d'événement, la notification d'événement est considérée comme étant une notification d'événement “heartbeat”. Ce service n’est pas confirmé.

#### 6.2.5.3.5.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 52.

**Tableau 52 – Paramètres du service Event notification**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Notifier ID | C | C (=) |
| Sequence Number | U | U (=) |
| Notification Time | U | U (=) |
| List of Event Messages | U | U (=) |
| List of Event Key Attributes | M | M (=) |
| List of Event Data types | C | C (=) |
| List of EventMessage Contents | C | C (=) |
| Reported Event Count | C | C (=) |
| Event Detection Time | C | C (=) |
| Event Data | C | C (=) |

**Argument**

L'argument contient les paramètres de la demande de service.

**Destination DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse de destination à laquelle la notification d'événement demandée est à envoyer.

**Source DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse source à partir de laquelle la notification d'événement indiquée est à envoyer.

**Notifier ID**

Ce paramètre conditionnel identifie le notificateur produisant la notification d'événement. Il est présent si l'AP a plus d'un notificateur défini pour lui.

**Sequence Number**

Ce paramètre facultatif est le numéro de séquence pour la notification d'événement. Il peut être utilisé pour des besoins de récupération de notification.

**Notification Time**

Ce paramètre facultatif est l'indicateur de temps pour la notification d'événement.

**List of Event Messages**

Ce paramètre facultatif spécifie la liste de messages d'événement qui sont à rapporter. Il peut contenir des messages issus d'un ou plusieurs objets Event, chacun contenant le même ensemble de paramètres (spécifiés par l'attribut Contained Message Type de l'objet 'Notifier'). Le contenu de chaque message est spécifié par son objet Event et il convient qu'il soit compatible avec celui spécifié pour l'objet "Notifier". Lorsque ce paramètre n'est pas présent, la notification d'événement est traitée comme une notification “heartbeat”.

**List of Event Key Attributes**

Ce paramètre identifie chacun des événements spécifiques acquittés par ce service.

**List of Event Data types**

Ce paramètre facultatif conditionnel indique le data type de chacun des paramètres Event Data (données d'événement). Ce paramètre ne peut être présent que si le paramètre Event Data est présent. Si le paramètre Event Data est présent, ce paramètre peut être présent, mais n'est pas tenu de l'être.

**List of Event Message Contents**

Ce paramètre facultatif conditionnel spécifie les messages d'événement générés par les objets Event.

**Reported Event Count**

Ce paramètre conditionnel rapporte le compte d'événements pour l'apparition rapportée. Ce paramètre est présent seulement s'il est défini pour le type de message de l'objet Event spécifié et si le type de message en question est pris en charge par le notificateur spécifié.

**Event Detection Time**

Ce paramètre facultatif rapporte l'heure de détection de l'événement. Ce paramètre est présent seulement s'il est défini pour le type de message de l'objet Event spécifié et si le type de message en question est pris en charge par le notificateur spécifié.

**Event Data**

Ce paramètre conditionnel spécifie les données utilisateur devant être incluses dans un message d'événement en plus de celles utilisées pour identifier l'apparition d'événement. Ce paramètre est présent seulement s'il est défini pour l'objet Event spécifié et s'il est pris en charge par le notificateur spécifié.

#### 6.2.5.3.5.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.5.3.6 Service Enable event list

#### 6.2.5.3.6.1 Vue d'ensemble du service

Ce service permet l'activation ou la désactivation de l'objet Event.

#### 6.2.5.3.6.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 53.

**Tableau 53 – Enable event list**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| List Of Key Attributes | M | M (=) | | |
| List Of Enable Flags | M | M (=) | | |
| Result(+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| List Of Enable Result | | | M | M (=) |
| List Of Error | | | C | C (=) |
| Result(–) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante 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 spécifiques au service relatifs à la demande du service.

**List of Key Attributes**

Ce paramètre identifie chaque événement par l'un de ses attributs-clés.

**List of Enable Flags**

Ce paramètre indique pour chaque événement si l'objet Objet correspondant est à activer (lorsque le fanion d'événement respectif est TRUE) ou être désactivé (lorsque le fanion d'événement respectif est FALSE).

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**List of Enable Result**

Ce paramètre liste de booléens indique, pour chaque événement qui devait être activé ou désactivé, si l'action Enable ou Disable avait réussi (Enable Result = TRUE) ou échoué (Enable Result = FALSE).

**List of Error**

Ce paramètre liste d'erreurs donne des informations relatives au type d'erreur survenue pour chaque événement dont l'action Enable ou Disable a échoué.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.5.3.6.3 Procédure de service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service. Le service Enable Eventlist est un service de "best effort".

#### 6.2.5.3.7 Service Notification recovery

##### 6.2.5.3.7.1 Vue d'ensemble du service

Ce service non confirmé est utilisé pour demander qu'un nombre spécifié de notifications d'événements retenues soit retourné. Les notifications sont retournées en utilisant le service de notification d'événement (Event Notification).

##### 6.2.5.3.7.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 54.

**Tableau 54 – Paramètres du service Notification recovery**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Notifier ID | M | M (=) |
| Number To Recover | U | U (=) |
| Sequence Number | U | U (=) |

**Argument**

L'argument contient les paramètres de la demande du service.

**Destination DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse de destination à laquelle la demande de récupération de notification indiquée est à envoyer.

**Source DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse source à partir de laquelle la demande de récupération de notification indiquée est à envoyer.

**Notifier ID**

Ce paramètre identifie le notificateur vers lequel ce service est dirigé.

**Number to recover**

Ce paramètre facultatif spécifie le nombre de notifications d'événements devant être envoyées de nouveau. Si ce nombre est supérieur au nombre réel stocké, seules notifications disponibles seront envoyées. Sa valeur par défaut est un.

**Sequence Number**

Ce paramètre facultatif spécifie le numéro de séquence de la plus ancienne notification d'événement devant être envoyée de nouveau. S'il n'est pas présent, la dernière notification envoyée est demandée. Il ne peut être utilisé que lorsque le notificateur spécifié utilise des numéros de séquence.

##### 6.2.5.3.7.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

La récupération est accomplie par l'AP répondeur en émettant de nouveau des demandes de service Event Notification contenant les messages de notification devant être récupérés.

#### 6.2.5.3.8 Service Get event summary

##### 6.2.5.3.8.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour demander des informations récapitulatives pour un objet Event spécifié.

##### 6.2.5.3.8.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 55.

**Tableau 55 – Paramètres du service Get event summary**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Event ID | M | M (=)) | | |
| Recovered Message Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Event Status | | | M | M (=) |
| Active | | | M | M (=) |
| Detected | | | M | M (=) |
| Enabled | | | M | M (=) |
| Acknowledgment Supported | | | M | M (=) |
| Acknowledged | | | M | M (=) |
| Object Revision | | | C | C (=) |
| Recovered Message | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Event ID**

Ce paramètre obligatoire identifie l'objet Event pour lequel le résumé d'événements est demandé. Un objet Event List (liste d'événements) ne peut pas être spécifié, car les éléments de la liste peuvent être changés dynamiquement par l'AP, rendant nécessaire l'identification de chaque événement dans la réponse.

**Recovered Message Requested**

Ce paramètre facultatif spécifie, lorsqu'il est TRUE, que le dernier message généré par l'événement est à retourner.

**Object Revision Requested**

Ce paramètre facultatif est utilisé pour demander que la valeur de l'attribut "Object Revision" (révision d'objet) soit retournée. Une valeur TRUE indique que la révision d'objet est

souhaitée. Une valeur FALSE indique que la révision d'objet n'est pas souhaitée. Si Object Revision est demandé, mais n'est pas pris en charge pour l'objet spécifié, le service échoue.

**Result (+)**

Cette sélection indique que la demande de service a réussi.

**Event Status**

Ce paramètre spécifie les valeurs de l'attribut statut d'événement. Il indique le statut de l'événement. Le statut est indiqué par une combinaison de paramètres booléens: active, detected, enabled, acknowledgment support et acknowledged.

**Detected**

Ce paramètre booléen indique, lorsqu'il est TRUE, que les conditions de détection d'une apparition d'événement sont vraies (TRUE).

**Active**

Ce paramètre booléen indique, lorsqu'il est TRUE, que les événements peuvent être détectés. S'il est FALSE, les apparitions d'événement ne sont pas détectées.

**Enabled**

Ce paramètre Boolean indique indique, lorsqu'il est TRUE, que les messages d'événement sont à rapporter au notificateur concerné à la suite de la détection d'apparitions d'événements. S'il est FALSE, il ne faut pas générer de message d'événement ni les rapporter lorsque des apparitions d'événement sont détectées.

**Acknowledgment Supported**

Ce paramètre Boolean indique, lorsqu'il est TRUE, que cet objet Event prend en charge l'acquittement de messages d'événement.

**Acknowledged**

Ce paramètre Boolean indique, lorsqu'il est TRUE, que tous les messages d'événement générés pour cet objet Event ont été acquittés. Cet attribut sera également TRUE si aucun message d'événement n'a été généré ou si l'objet Event ne prend pas en charge l'acquittement. La valeur FALSE indique que l'objet Event attend un acquittement.

**Recovered Message**

Ce paramètre conditionnel est présent si la demande de service spécifiait que la récupération de message était demandée. Lorsqu'il est présent, ce paramètre spécifie le message d'événement demandé.

**Object Revision**

Ce paramètre conditionnel spécifie la révision d'objet relative à l'objet spécifié. Il est présent si la révision d'objet a été demandée.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.5.3.8.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Get Event Summary est un service "tout ou rien" qui opère à travers une file d'attente ou un buffer. "Tout ou rien" signifie que le service réussit seulement si le résumé demandé est retourné.

#### 6.2.5.3.9 Service Get event summary list

##### 6.2.5.3.9.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour demander des informations récapitulatives pour une liste d'objets Event spécifiés.

##### 6.2.5.3.9.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 56.

**Tableau 56 – Paramètres du service Get event summary list**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Summary Requests | M | M (=)) | | |
| Requested Events | M | M (=)) | | |
| All Events | S | S (=) | | |
| List Of Events | S | S (=) | | |
| Recovered Message Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Event Summaries | | | M | M (=) |
| List Of Event ID | | | C | C (=) |
| List Of Access Result | | | M | M (=) |
| List Of Response | | | M | M (=) |
| Error Status | | | S | S (=) |
| Event Summary | | | S | S (=) |
| Event Status | | | M | M (=) |
| Active | | | M | M (=) |
| Detected | | | M | M (=) |
| Enabled | | | M | M (=) |
| Acknowledgment Supported | | | M | M (=) |
| Acknowledged | | | M | M (=) |
| Object Revision | | | C | C (=) |
| Recovered Message | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Summary Requests**

Ce paramètre obligatoire spécifie une liste d'événements, et les informations récapitulatives relatives à chacun de ceux-ci, qui est demandée. Chaque entrée de la liste peut être

identifiée soit par un objet Event, soit par un objet Event List. Chaque entrée de la liste identifie également les informations récapitulatives qui sont demandées. Si une entrée identifie une liste d'événements, les informations récapitulatives demandées s'appliquent à tous les événements contenus dans la liste d'événements.

**Requested Events**

Ce paramètre obligatoire indique que les résumés sont demandés pour tous les événements ou il contient une liste qui identifie les objets Event pour lesquels les résumés d'événements sont demandés.

**All Events**

Ce paramètre de type sélection indique que des résumés sont demandés pour tous les événements.

**List of Events**

Ce paramètre de type sélection est une liste qui contient les identificateurs des événements/ou listes d'événements pour lesquel(le)s des résumés d'événements sont demandés. Lorsqu'il est sélectionné, il convient que la liste contienne au moins une entrée.

**Recovered Message Requested**

Ce paramètre facultatif spécifie, lorsqu'il est TRUE, que le dernier message généré par l'événement est à retourner.

**Object Revision Requested**

Ce paramètre facultatif est utilisé pour demander que la valeur de l'attribut "Object Revision" (révision d'objet) soit retournée. Une valeur TRUE indique que la révision d'objet est souhaitée. Une valeur FALSE indique que la révision d'objet n'est pas souhaitée. Si Object Revision est demandé, mais n'est pas pris en charge pour l'objet spécifié, le service échoue.

**Result (+)**

Cette sélection indique que la demande de service a réussi.

**Event Summaries**

Ce paramètre spécifie une réponse pour chaque événement demandé. L'ordre des informations est le même que l'ordre dans lequel les objets étaient référencés dans la demande de service.

**List of Event ID**

Ce paramètre conditionnel identifie chaque événement pour lequel un résumé d'événement ou un statut d'erreur est retourné.

**List of Access Result**

Cette liste de paramètres booléens fournit pour chaque événement pour lequel un résumé d'événements avait été demandé une information indiquant si l'accès a réussi (Access Result = TRUE) ou non (Access Result = FALSE).

**List of Response**

Ce paramètre spécifie un indicateur de statut d'erreur ou le résumé d'événements pour chaque événement demandé. L'ordre des informations est le même que l'ordre dans lequel les objets étaient référencés dans la demande de service.

**Error Status**

Ce paramètre de type sélection spécifie si la condition qui déclenche les rapports relatifs à l'événement est active. Une valeur TRUE indique que la condition sous-

jacente qui déclenche l'événement est active. Une valeur FALSE indique que la condition sous-jacente qui déclenche l'événement n'est pas active.

**Event Summary**

Ce paramètre de type sélection spécifie les informations récapitulatives pour l'événement spécifié.

**Event Status**

Ce paramètre spécifie les valeurs de l'attribut statut d'événement. Il indique le statut de l'événement. Le statut est indiqué par une combinaison de paramètres booléens: active, detected, enabled, acknowledgment support et acknowledged.

**Détecté**

Ce paramètre booléen indique, lorsqu'il est TRUE, que les conditions de détection d'une apparition d'événement sont vraies (TRUE).

**Active**

Ce paramètre booléen indique, lorsqu'il est TRUE, que les événements peuvent être détectés. S'il est FALSE, les apparitions d'événement ne sont pas détectées.

**Enabled**

Ce paramètre Boolean indique, lorsqu'il est TRUE, que les messages d'événement sont à rapporter au notificateur concerné à la suite de la détection d'apparitions d'événements. S'il est FALSE, il ne faut pas générer de de message d'événement ni les rapporter lorsque des apparitions d'événements sont détectées.

**Acknowledgment Supported**

Ce paramètre Boolean indique, lorsqu'il est TRUE, que cet objet Event prend en charge l'acquittement de messages d'événement.

**Acknowledged**

Ce paramètre Boolean indique, lorsqu'il est TRUE, que tous les messages d'événement générés pour cet objet Event ont été acquittés. Cet attribut sera également TRUE si aucun message d'événement n'a été généré ou si l'objet Event ne prend pas en charge l'acquittement. La valeur FALSE indique que l'objet Event attend un acquittement.

**Recovered Message**

Ce paramètre conditionnel est présent si la demande de service spécifiait que la récupération de message était demandée. Lorsqu'il est présent, ce paramètre spécifie le message d'événement demandé.

**Object Revision**

Ce paramètre conditionnel spécifie la révision d'objet relative à l'objet spécifié. Il est présent si la révision d'objet a été demandée.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.5.3.9.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Get Event Summary List est un service " best effort" qui opère à travers une file d'attente ou un buffer. "Best effort" signifie que le service réussit si au moins résumé d'événements est retourné. Si l'utilisateur n'est pas capable de lire au moins l'une des valeurs, le service échoue et l'utilisateur produit une primitive "response (-)" du service Get Event Summary List en indiquant la cause.

#### 6.2.5.3.10 Service Query event summary list

##### 6.2.5.3.10.1 Vue d'ensemble du service

Ce service confirmé fournit le moyen pour une application de demander que tous les événements, une liste d'événement ou un groupe d'événements soient explorés conformément au critère fourni. La liste de critères valides pris en charge par le processus application répondeur est définie par l'attribut List of Event Summary Selection Criteria de l'AP répondeur. Un critère pris en charge peut définir soit une caractéristique individuelle d'événement, soit une combinaison de caractéristiques d'événement.

NOTE L'attribut critères de sélection défini pour un événement peut reposer sur une caractéristique d'événement telle que son statut, sur le fait de savoir s'il est enabled (activé) ou disabled (désactivé) ou sur le fait de savoir si, oui ou non, il a été acquitté. Il est également possible de spécifier des critères définis par un utilisateur tels que les événements relatifs à une unité opérationnelle particulière ou à une pièce d'équipement.

##### 6.2.5.3.10.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 57.

**Tableau 57 – Paramètres du service Query event summary list**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Events to Search | M | M (=)) | | |
| All Events | S | S (=) | | |
| List Of Events | S | S (=) | | |
| Selection Criterion | M | M (=) | | |
| Recovered Message Requested | U | U (=) | | |
| Object Revision Requested | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Event Summaries | | | M | M (=) |
| List Of Event ID | | | C | C (=) |
| List Of Access Result | | | M | M (=) |
| List Of Response | | | M | M (=) |
| Event Status | | | M | M (=) |
| Active | | | M | M (=) |
| Detected | | | M | M (=) |
| Enabled | | | M | M (=) |
| Acknowledgment Supported | | | M | M (=) |
| Acknowledged | | | M | M (=) |
| Object Revision | | | C | C (=) |
| Recovered Message | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |

NOTE Voir la Note en 3.8.4.3.

### Argument

L'argument contient les paramètres de la demande du service.

### Events to Search

Ce paramètre identifie les objets Event qui sont à explorer. Les événements sont identifiés en spécifiant tous les événements, un groupe d'événements ou une liste d'événements.

#### All Events

Ce paramètre de type sélection indique que tous les événements sont à explorer.

#### List of Events

Ce paramètre de type sélection spécifie les événements qui sont à explorer. Les événements peuvent être spécifiés comme étant une liste d'objets Event, une liste d'objets Event List, ou une combinaison des deux. Lorsqu'il est sélectionné, il convient que la liste contienne au moins une entrée.

**Selection Criterion**

Ce paramètre identifie le critère à utiliser dans le processus de sélection. Il convient que la valeur utilisée soit contenue dans l'attribut List of Event Summary Selection Criteria de l'AP contenant les événements devant être explorés.

**Recovered Message Requested**

Ce paramètre facultatif spécifie, lorsqu'il est TRUE, que le dernier message généré par l'événement est à retourner.

**Object Revision Requested**

Ce paramètre facultatif est utilisé pour demander que la valeur de l'attribut "Object Revision" (révision d'objet) soit retournée. Une valeur TRUE indique que la révision d'objet est souhaitée. Une valeur FALSE indique que la révision d'objet n'est pas souhaitée. Si Object Revision est demandé, mais n'est pas pris en charge pour l'objet spécifié, le service échoue.

**Result (+)**

Cette sélection indique que la demande de service a réussi.

**Event Summaries**

Ce paramètre spécifie les informations récapitulatives pour chaque événement sélectionné.

**List of Event ID**

Cet attribut conditionnel est présent lorsque le critère de sélection n'est pas “none” ou lorsque le critère de sélection est “none” et les événements spécifiés sont “all” ou “list of events”.

**List of Access Result**

Cette liste de paramètres booléens fournit pour chaque événement pour lequel un résumé d'événements est retourné une information indiquant si l'accès a réussi (Access Result = TRUE) ou non (Access Result = FALSE).

**List of Response**

Ce paramètre spécifie un indicateur de statut d'erreur ou le résumé d'événements pour chaque résumé d'événement retourné. L'ordre des informations est le même que l'ordre dans lequel les objets étaient référencés dans la demande de service.

**Event status**

Ce paramètre spécifie, lorsqu'il est TRUE, que les conditions de détection d'un événement sont actuellement vraies (TRUE). Ce paramètre a la valeur FALSE si les conditions sont FALSE ou si l'objet Event n'est pas actif.

**Active**

Ce paramètre spécifie, lorsqu'il est TRUE, que la détection d'événement est active pour l'événement. Lorsqu'il est actif, l'objet Event détecte l'apparition d'événement. Lorsqu'il est inactif, il ne le fait pas.

**Enabled**

Ce paramètre spécifie la valeur de l'attribut Enabled de l'objet Event identifié.

**Unacknowledged**

Ce paramètre, lorsqu'il est TRUE, spécifie que l'objet Event n'a pas d'occurrences d'événement en souffrance pour lesquelles il attend des acquittements ou qu'il ne prend pas en charge l'acquittement. Si l'Event attend des acquittements, ce paramètre a une valeur FALSE.

**Recovered Message**

Ce paramètre conditionnel est présent si la demande de service spécifiait que la récupération de message était demandée. Lorsqu'il est présent, ce paramètre spécifie le message d'événement demandé.

**Object Revision**

Ce paramètre conditionnel est présent si la demande de service spécifiait que la révision d'objet était demandée. Lorsqu'il est présent, ce paramètre facultatif spécifie la révision d'objet de l'objet Event sélectionné.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.5.3.10.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

Le service Query Event Summary est un service "de best effort" qui opère à travers une file d'attente ou un buffer. "Best effort" signifie que le service réussit si au moins résumé d'événements est retourné. Si l'utilisateur n'est pas capable d'obtenir un résumé pour au moins un événement, le service échoue et l'utilisateur produit une primitive "response (-)" du service Query Event Summary en indiquant la cause.

### 6.2.6 ASE Load Region

#### 6.2.6.1 Vue d'ensemble

Un objet Load Region représente une zone de mémoire non structurée dont le contenu peut être téléchargé vers l'amont, c'est-à-dire téléversé (lu) ou téléchargé vers l'aval, c'est-à-dire téléchargé (écrit). Dans ce contexte, "Non structuré" signifie que la zone de mémoire est représentée seulement comme étant une séquence ordonnée d'octets. Aucune autre structure n'est apparente.

Les régions de chargement peuvent représenter une zone non nommée de mémoire volatile (telle que celle mise en œuvre par une mémoire d'ordinateur dynamique) ou un objet nommé de mémoire non volatile (tel qu'un fichier). Le contenu d'une région de chargement est appelé "load image" (image du chargement). Les images de chargement peuvent contenir des programmes ou des données. Le transfert d'une image de chargement en direction ou en provenance d'une région de charge est accompli en utilisant le processus de chargement.

Pour maintenir l'intégrité, un seul processus de chargement pour une région de chargement est permis à la fois. L'attribut Load Region State de la région de chargement est utilisé pour indiquer si la région de chargement est vide, en train d'être téléchargé vers l'aval ou chargé (pour une liste complète des états, voir la définition de l'attribut). Les régions de chargement peuvent être effacées en utilisant le service Discard.

L'attribut Upload State indique également l'état d'avancement d'un téléchargement vers l'amont (ou téléversement). Cet attribut est défini pour séparer l'état du contenu de la région de chargement du fonctionnement consistant à avoir le contenu téléchargé vers l'amont (déversé).

Ce modèle de région de chargement fournit des services pour lancer le téléchargement ou le téléversement de l'une de ses régions de chargement. Il identifie la région de chargement et s'il faut la téléverser ou télécharger comme paramètre de la demande. Un troisième paramètre indique si le transfert des segments d'image de chargement est à accomplir en utilisant le service Pull Segment ou le service Push Segment.

NOTE Les services Push Segment et Pull Segment n'ont aucune relation avec les types d'AREP Push et Pull.

Les segments d'image de chargement sont tirés par le fait que le destinataire de l'image de chargement émet des demandes les concernant. L'AP qui contient l'image de chargement répond en retournant le segment demandé, en indiquant avec un paramètre le moment où le segment final de l'image de chargement a été retourné.

Les segments d'image de chargement sont poussés par le fait que l'AP contenant l'image de chargement envoie des segments individuels vers le destinataire et attende une réponse qui indique si, oui ou non, le segment a été reçu. Dans ce cas, l'expéditeur indique le moment où le dernier segment a été transféré.

Après que le dernier segment a été reçu, l'AP qui a lancé le transfert met fin au processus de chargement en émettant une demande d'arrêt à l'AP distant. La demande d'arrêt peut également être utilisée par l'un ou l'autre AP si l'AP détermine que le processus de chargement ne peut pas être achevé avec succès.

#### 6.2.6.2 Spécification du modèle de Load region

##### 6.2.6.2.1 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE LOAD REGION** |
| **CLASS:** | LOAD REGION | | |
| **CLASS ID:** | | | 2 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut: | Load Region Size |
| 2 | (m) | Attribut: | Access Privilege |
| 2.1 | (m) | Attribut: | Password |
| 2.2 | (m) | Attribut: | Access Groups |
| 2.3 | (m) | Attribut: | Access Rights |
| 3 | (m) | Attribut: | Local Address |
| 4 | (m) | Attribut: | Load Region State |
| 5 | (m) | Attribut: | Upload State |
| 6 | (m) | Attribut: | Number of Related Objects In Use |
| 7 | (o) | Attribut: | List of Related Objects |
| 7.1 | (m) | Attribut: | ClassID |
| 7.2 | (m) | Attribut: | Numeric ID |
| 7.2 | (m) | Attribut: | In Use Flag (TRUE, FALSE) |
| 8 | (o) | Attribut: | Contents Size |
| 9 | (o) | Attribut: | Load Image Name |
| 10 | (o) | Attribut: | Fixed Length Segments ( TRUE/FALSE) Default: FALSE |
| 11 | (c) | Contrainte: | Fixed Length Segments = TRUE |
| 11.1 | (o) | Attribut: | Fixed Segment Length |
| 12 | (o) | Attribut: | Intersegment Request Timeout |
| 13 | (o) | Attribut: | Sharable |
| 14 | (o) | Attribut: | Local Detail |
| **SERVICES:** | | | |
| 1 | (m) | Ops Service: | Initiate Load |
| 2 | (o) | Ops Service: | Push Segment |
| 3 | (o) | Ops Service: | Pull Segment |
| 4 | (m) | Ops Service: | Terminate Load |
| 5 | (o) | Ops Service: | Discard |

##### 6.2.6.2.2 Attributs

**Load Region Size**

Cet attribut spécifie la taille maximale de l'objet Load Region en octets.

**Access Privilege**

Cet attribut spécifie les contrôles d'accès définis pour cette région de chargement. Il se compose des éléments suivants:

**Password**

Cet attribut spécifie le mot de passe pour les droits d'accès. Sa valeur est null s'il n'est pas utilisé.

**Access Groups**

Cet attribut identifie lesquels des huit groupes d'accès définis par un utilisateur sont définis pour la région de chargement. Plus d'un groupe d'accès peut être défini.

**Access Rights**

Cet attribut définit le type d'accès défini pour la région de chargement. Les valeurs valides sont:

- Droit d'utilisation dans un FI pour les groupes d'accès
- Droit d'écrire pour les groupes d'accès
- Droit de lire pour les groupes d'accès
- Droit d'utilisation dans un FI pour le mot de passe enregistré
- Droit d'écrire pour le mot de passe enregistré
- Droit de lire pour le mot de passe enregistré
- Droit d'utilisation dans un FI pour tous les partenaires de communication
- Droit d'écrire pour tous les partenaires de communication
- Droit de lire pour tous les partenaires de communication

**Local Address**

Cet attribut est l'adresse localement pertinente de la région de chargement. La valeur hexadécimale FFFFFFFF indique qu'aucune adresse locale n'est disponible.

**Load Region State**

Cet attribut spécifie l'état de l'objet Load Region. Les valeurs de cet attribut sont:

NOT DOWNLOADABLE Cet état indique que la Load Region n'est pas capable d'être téléchargée vers l'aval.

DOWNLOADABLE Cet état indique que la Load Region est vide, mais est capable d'être téléchargée vers l'aval.

DOWNLOADING Cet état indique que la Load Region est vide, mais est capable d'être téléchargée vers l'aval

DOWNLOAD FAILURE Cet état transitoire indique que la séquence de téléchargement vers l'aval a échoué, mais n'a pas encore été terminée.

DOWNLOAD SUCCESS Cet état transitoire indique que la séquence de téléchargement vers l'aval a réussi, mais n'a pas encore été terminée.

LOADED Cet état indique que la séquence de téléchargement vers l'aval s'est terminée avec succès.

IN-USE Cet état indique que la Load Region est chargée et est actuellement utilisée.

Contraintes sur l'attribut Download State:

Un objet Load Region dont le contenu est stocké dans une mémoire non effaçable ne peut pas prendre en charge des transferts en téléchargement vers l'aval. Ses valeurs d'état valides sont LOADED et IN USE.

Un objet Load Region dont le contenu est stocké dans une mémoire effaçable prend en charge des transferts en téléchargement vers l'aval. Ses valeurs d'état valides sont DOWNLOADABLE, DOWNLOADED, DOWNLOADING, DOWNLOADING_FAILURE, DOWNLOADING_SUCCESS, LOADED et IN_USE,

NOTE L'état NOT DOWNLOADABLE est réservé pour les Load Region dont le contenu est stocké dans une mémoire effaçable, mais qui n'ont pas rencontré un état temporaire qui ne leur permet pas d'être téléchargés vers l'aval.

**Upload State**

Cet attribut spécifie l'état du processus de téléchargement vers l'amont (téléversement) de la région de chargement.

NOT UPLOADABLE Cet état indique que la Load Region n'est pas capable d'être téléchargée vers l'amont (téléversée).

UPLOADABLE Cet état indique que la Load Region est capable d'être téléchargée vers l'amont (téléversée).

UPLOADING Cet état indique que la séquence de téléchargement vers l'amont (téléversement) a été lancée.

COMPLETING UPLOAD Cet état transitoire indique que la séquence de téléchargement vers l'amont a été achevée, mais n'a pas encore été arrêtée.

Contraintes sur l'attribut Upload State:

Un objet Load Region dont le contenu est stocké dans une mémoire effaçable ou qui ne prend pas en charge des transferts en téléchargement vers l'amont. Ses valeurs d'état valides sont UPLOADABLE, UPLOADING et COMPLETING UPLOADING

NOTE L'état NOT UPLOADABLE est réservé aux Load Region dont le contenu est stocké en mémoire effaçable, mais ont rencontré un état temporaire qui ne leur permet pas d'être téléchargées vers l'amont (téléversées).

**Number of Related Objects in Use**

Cet attribut indique le nombre d'objets Action ou d'objets Function Invocation connexes qui utilisent actuellement le contenu de l'objet Load Region.

Cet attribut est incrémenté chaque fois qu'un objet Function Invocation connexe entre dans l'état RUNNING ou qu'un objet Action connexe est invoqué.

Cet attribut est décrémenté chaque fois qu'un objet Function Invocation connexe quitte l'état RUNNING ou qu'un objet Action connexe achève son exécution.

Si l'attribut sharable = TRUE, plusieurs objets Function Invocation et objets Action peuvent partager le contenu de Load Region; autrement, un seul objet Function Invocation ou objet Action peut être attaché et utiliser le contenu de l'objet.

**List of Related Objects**

Cet attribut facultatif spécifie les objets Function Invocation et Action qui partagent le contenu de la Load Region. Chaque objet Function Invocation ou objet Action est identifié en spécifiant son identificateur de classe et son identificateur numérique.

La liste est respectivement augmentée/diminuée chaque fois qu'un objet Function Invocation ou objet Action connexe est créé ou supprimé.

**Class ID**

Cet attribut est l'identificateur de classe de l'objet connexe.

**Numeric ID**

Cet attribut est l'identificateur numérique de l'objet connexe.

**In Use Flag**

Cet attribut est TRUE si l'objet connexe est un objet Function Invocation dans l'état RUNNING ou si l'objet connexe est un objet Action qui s'exécute actuellement.

**Contents Size**

Cet attribut facultatif spécifie la longueur des données contenues dans la Load Region. La longueur est indiquée en octets. Si l'état de la région de chargement est DOWNLOADABLE, la valeur de cet attribut est zéro.

**Load Image Name**

Cet attribut facultatif spécifie le nom de l'image de chargement contenue dans la région de chargement.

**Fixed-Length Segments**

Cet attribut facultatif spécifie, lorsqu'il est TRUE, que la région de chargement prend en charge le téléchargement vers l'amont et vers l'aval des segments seulement de longueur fixe. La valeur par défaut pour cet attribut est FALSE.

**Fixed-Segment Length**

Cet attribut conditionnel spécifie la longueur en octets des segments de longueur fixe téléchargés vers l'amont vers cette région de chargement et téléchargés vers l'aval à partir de cette région de chargement. L'attribut est présent lorsque l'attribut Fixed Length Segments est TRUE.

**Intersegment Request Timeout**

Cet attribut facultatif spécifie, en secondes la période de temporisation pour recevoir des demandes de segment.

**Sharable**

Cet attribut indique, lorsqu'il est TRUE, que le contenu de l'objet Load Region peut être utilisé par plusieurs objets Function Invocation/Action. La valeur par défaut pour cet attribut est TRUE.

**Local Detail**

Cet attribut spécifie des informations complémentaires, telles qu'une liste de fonctionnalités et/ou le nombre d'octets devant être transféré(e).

#### 6.2.6.2.3 Services

**Initiate Load**

Ce service est utilisé par une application pour lancer le téléchargement (vers l'aval ou vers l'amont) d'une Load Region. Un paramètre du service indique si l'image de chargement sera transférée dans la Load Region en utilisant le service Pull Segment ou de la Load Region en utilisant le service Push Segment.

**Push Segment**

Ce service facultatif est utilisé pour transférer les segments de chargements dans les primitives "request"/"indication". Bien que ce service soit facultatif, au moins l'un des services de transfert de segments, Push Segment ou Pull Segment, est requis.

**Pull Segment**

Ce service facultatif est utilisé pour demander qu'un segment de chargement soit retourné dans les primitives "response"/"confirmation". Bien que ce service soit facultatif, au moins l'un des services de transfert de segments, Push Segment ou Pull Segment, est requis.

**Terminate Load**

Ce service est utilisé par un AP pour mettre fin au processus local.

**Discard**

Ce service facultatif est utilisé pour effacer le contenu de la région de chargement.

#### 6.2.6.3 Spécification de services de Load region

##### 6.2.6.3.1 Services pris en charge

Ce paragraphe contient la définition de services qui sont propres à cet ASE. Les services définis pour cet ASE sont:

Initiate Load

Terminate Load

Push Segment

Pull Segment

Discard

##### 6.2.6.3.2 Service Initiate load

###### 6.2.6.3.2.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour demander le téléchargement (vers l'aval ou vers l'amont) d'une région de chargement. Un paramètre du service indique si l'image de chargement est à transférer par le demandeur en utilisant le service Pull Segment ou le service Push Segment. Lorsqu'un téléchargement vers l'aval est bien demandé pour une région de chargement dans un AP distant (identifié par l'attribut-clé de la région de chargement appelée) qui n'existe pas encore, l'AP répondeur peut soit créer dynamiquement la région de chargement et retourner une réponse positive, soit retourner une réponse négative qui indique que la région de chargement n'existe pas.

###### 6.2.6.3.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 58.

**Tableau 58 – Paramètres du service Initiate load**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Region Key Attribute | M | M (=) | | |
| Calling | S | S (=) | | |
| Appelé | S | S (=) | | |
| Load Type | M | M (=) | | |
| Load Service To Use | M | M (=) | | |
| Load Service Initiator | M | M (=) | | |
| Additional Information | U | U (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Additional Detail | | | U | U (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Called Size | | | C | C (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande de service.

**Load Region Key Attribute**

Ce paramètre identifie la région de chargement devant être téléchargée vers l'aval ou téléchargée vers l'amont.

**Calling**

Ce paramètre de type sélection identifie la région de chargement du demandeur. Il est sélectionné si le demandeur est le serveur.

**Called**

Ce paramètre de type sélection identifie la région de chargement du répondeur. Il est sélectionné si le demandeur est le client.

**Load Type**

Ce paramètre spécifie si le chargement est un UPLOAD (sortant de la région de chargement) ou un DOWNLOAD (pénétrant dans la région de chargement).

**Load Service to Use**

Ce paramètre spécifie que la région de chargement est à transférer à partir de l'AP distant en utilisant soit le service Pull Segment, soit le service Push Segment.

**Load Service Initiator**

Ce paramètre spécifie, lorsqu'il est TRUE, que le service de chargement identifié par le précédent paramètre est à lancer par l'AP lançant ce service. Lorsqu'il est FALSE, il faut que l'AP distant lance le service de chargement.

**Additional Information**

Ce paramètre facultatif spécifie des informations complémentaires, telles qu'un nom de fichier et/ou le nombre d'octets devant être transféré(e). Il peut être présent si le demandeur est le serveur.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Additional Detail**

Ce paramètre facultatif spécifie des informations complémentaires, telles qu'une liste de fonctionnalités et/ou le nombre d'octets devant être transféré(e). Il peut être présent si le répondeur est le serveur.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Called Size**

Ce paramètre conditionnel spécifie la taille maximale en octets que l'AP répondeur est préparé à transférer. Il est présent si l'erreur était due à un problème lié à la taille.

#### 6.2.6.3.2.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.6.3.3 Service Terminate load

#### 6.2.6.3.3.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour mettre fin au processus local. Il peut être utilisé à la suite d'achèvement réussi du chargement ou pour abandonner un chargement en cours. Il peut être demandé par l'une ou l'autre des points d'extrémité.

Si le processus de chargement a été parachevé, il est toujours émis par l'AP qui a émis le service de lancement (Initiate) qui a démarré le processus de chargement. Si le processus de chargement est en cours d'abandon,

a) la région de chargement envoyant l'image retourne à l'état Upload State mis à UPLOADABLE ;

b) la région de changement recevant l'image supprime de sa région de chargement le chargement inachevé et retourne à l'état DOWNLOADABLE.

#### 6.2.6.3.3.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 59.

**Tableau 59 – Paramètres du service Terminate load**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Region Key Attribute | M | M (=) | | |
| Load Type | M | M (=) | | |
| Load Service Used | M | M (=) | | |
| Terminate Reason | C | C (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Terminate Reason | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Load Region Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de la Load Region pour laquelle le chargement est en cours d'arrêt.

**Load Type**

Ce paramètre indique le type de chargement qui est en cours d'arrêt. Les valeurs valides sont UPLOAD et DOWNLOAD.

**Load Service Used**

Ce paramètre indique que le type de service de chargement utilisé pour transférer des segments de chargement. Les valeurs valides sont PUSH et PULL.

**Terminate Reason**

Ce paramètre conditionnel indique le succès ou l'échec du processus de chargement. Il est présent si le serveur émet la demande d'arrêt.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Terminate Reason**

Ce paramètre conditionnel indique le succès ou l'échec du processus de chargement. Il est présent si le client émet la demande d'arrêt pour mettre fin à un téléchargement vers l'aval.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.6.3.3.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

#### 6.2.6.3.4 Service Push segment

##### 6.2.6.3.4.1 Vue d'ensemble du service

Ce service confirmé est utilisé après que le processus de chargement a été lancé pour transférer un seul segment d'image de chargement dans ses primitives "request" et "indication". Les primitives "response" et "confirmation" sont utilisées pour acheminer le succès ou l'échec du transfert de segment. Ce service peut être utilisé pour prendre en charge des téléchargements vers l'amont et vers l'aval.

##### 6.2.6.3.4.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 60.

**Tableau 60 – Paramètres du service Push segment**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Type | M | M (=) | | |
| Load Region Key Attribute | M | M (=) | | |
| Segment Number | U | U (=) | | |
| Load Data | M | M (=) | | |
| More Follows | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Load Type**

Ce paramètre spécifie si le chargement est un UPLOAD (hors de la région de chargement) ou un DOWNLOAD (dans la région de chargement).

**Load Region Key Attributes**

Ce paramètre spécifie l'un des attributs-clés de la Load Region dont l'image est à transférer.

NOTE La région de chargement est locale si ce service est utilisé pour un téléchargement vers l'amont. La région de chargement est distante si ce service est utilisé pour un téléchargement vers l'aval.

**Segment Number**

Ce paramètre facultatif identifie le numéro du segment qui est transféré. Les numéros de segment sont des nombres entiers dans l'ordre croissant commençant par la valeur un (1).

**Load Data**

Ce paramètre spécifie les données devant être chargées.

**More Follows**

Ce paramètre indique si, oui ou non, des données complémentaires restent à émettre.

**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.

#### 6.2.6.3.4.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.6.3.5 Service Pull segment

#### 6.2.6.3.5.1 Vue d'ensemble du service

Ce service confirmé est utilisé après que le processus de chargement ait été lancé pour demander qu'un segment image du chargement spécifié soit retourné dans les primitives "response" et "confirmation". Ce service peut être utilisé pour prendre en charge des téléchargements vers l'amont et vers l'aval.

#### 6.2.6.3.5.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 61.

**Tableau 61 – Paramètres du service Pull segment**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Type | M | M (=) | | |
| Load Region Key Attribute | M | M (=) | | |
| Segment Number | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Load Data | | | M | M (=) |
| More Follows | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Load Type**

Ce paramètre spécifie si le chargement est un UPLOAD (hors de la région de chargement) ou un DOWNLOAD (dans la région de chargement).

**Load Region Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de la Load Region dont l'image est à transférer.

NOTE La région de chargement est distante si ce service est utilisé pour un téléchargement vers l'amont. La région de chargement est locale si ce service est utilisé pour un téléchargement vers l'aval.

**Segment Number**

Ce paramètre facultatif identifie le numéro du segment qui est demandé. Les numéros de segment sont des nombres entiers dans l'ordre croissant commençant par la valeur un (1).

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Load Data**

Ce paramètre spécifie les données devant être téléchargées (vers l'aval ou vers l'amont).

**More Follows**

Ce paramètre indique si, oui ou non, des données complémentaires restent à émettre.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.6.3.5.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.6.3.6 Service Discard

#### 6.2.6.3.6.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour demander que le contenu d'une région de chargement soit rejeté.

#### 6.2.6.3.6.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 62.

**Tableau 62 – Paramètres du service Discard**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Load Region Key Attribute | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

L'argument contient les paramètres de la demande du service.

**Load Region Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de l'objet Load Region dont le contenu est à rejeter.

**Result (+)**

Cette sélection indique que la demande de service a réussi.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

#### 6.2.6.3.6.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.6.4 Diagrammes d'états de Load Region

#### 6.2.6.4.1 Généralités

La présente norme définit les diagrammes d'états pour trois types de transferts.

Pull_Uploading: Le contenu de la région de chargement est récupéré d'un objet Load Region, segment par segment, par un AP client.

Pull_Downloading: Une image de chargement contenue dans un AP client est copiée vers l'objet Load Region, segment par segment, à la demande de l'AP serveur qui contient la région de chargement. Autrement dit, l'AP serveur demande au client d'envoyer les segments de l'image de chargement. Cette sorte de transfert est autorisée pour les régions de chargement effaçables. Son diagramme d'états et sa table de transitions donnent des informations complémentaires à un contenu de région de chargement non effaçable.

Push_Downloading: Une image de chargement contenue dans un AP client est envoyée vers l'objet Load Region, segment par segment, par l'AP client. À réception de chaque segment, l'AP serveur qui contient la région de chargement la copie vers la

région de chargement. Cette sorte de transfert est autorisée pour les régions de chargement effaçables. Son diagramme d'états et sa table de transitions donnent des informations complémentaires au contenu de région de chargement non effaçable.

Les autres sortes de transfert ne relèvent pas du domaine d'application de la présente norme.

#### 6.2.6.4.2 Diagrammes d'états de Pull upload (téléversement en mode Pull)

##### 6.2.6.4.2.1 Description

Cette sorte de transfert utilise les services suivants: Initiate Load, Pull Segment et Terminate Load

Un serveur peut lancer un transfert de téléchargement vers l'amont. Dans ce cas, il émet une primitive de service Initiate_Load.Req vers un client. Cette primitive dit au client de lancer le transfert de téléchargement vers l'amont pour une région de chargement spécifiée. Une fois le client a tiré le contenu, il émet une primitive Initiate_Load.Rsp(+/-) vers le serveur.

Un client peut lancer un transfert de téléchargement vers l'amont. Dans ce cas, il émet une primitive Initiate_Load.Req vers un serveur. Cette primitive dit au serveur que le contenu d'un objet Load Region spécifié sera tiré. Une fois que le client a tiré le contenu, il termine le transfert en émettant une primitive Terminate_Load.Req vers le serveur.

##### 6.2.6.4.2.2 Ordonnancement des primitives de service

Le Tableau 63 résume l'ordonnancement des primitives de service. Dans le tableau, T1 et T2 montrent différentes options pour lancer le téléchargement vers l'amont, l'une par l'AP serveur AP (T1) et l'une par l'AP client (T2).

**Tableau 63 – Ordonnancement des primitives de service de "Pull upload" (téléchargement vers l'amont en mode Pull)**

| **Utilisateur de la FAL (Client)** | | **Utilisateur de la FAL (Serveur) Propriétaire de la région de chargement appelée** |
|---|---|---|
| Initiate_Load.Ind | ←----(T 1)------ | Initiate_Load.Req |
| Initiate_Load.Req | -------(T2)-----→ | Initiate_Load.Ind (Transfert lancé) |
| Initiate_Load.Cnf | ←----(T2)------- | Initiate_Load.Rsp |
| Pull_Segment.Req | -------(T3)-----→ | Pull_Segment.Ind (Segments transférés) |
| Pull_Segment.Cnf | ←----(T3)------- | Pull_Segment.Rsp |
| | | |
| Terminate_Load.Req | -------(T4)-----→ | Terminate_Load.Ind (Transfert terminé) |
| Terminate_Load.Cnf | ←----(T4)------- | Terminate_Load.Rsp |
| Initiate_Load.Rsp | -------(T1)-----→ | Initiate_Load.Cnf |

##### 6.2.6.4.2.3 Contraintes sur les valeurs des paramètres de service

Le Tableau 64 résume les contraintes sur les paramètres de service.

**Tableau 64 – Contraintes sur les paramètres de service "Pull upload" (téléversement en mode Pull)**

| Numéro de transition | Contraintes sur les paramètres |
|---|---|
| T1 | Initiate_Load<br>Calling Load Region Key attribute selected,<br>Load Type : = upload,<br>Service To Use : = Pull_Segment,<br>Load Service Initiator : = FALSE |
| T2 | Initiate_Load<br>Called Load Region Key Attribute selected,<br>Load Type : = upload<br>Service To Use : = Pull_Segment<br>Load Service Initiator : = TRUE |
| T3 | Pull_Segment<br>Load Type : = upload,<br>Segment Number : = ordered 1 to n if any<br>More Follows : = TRUE/FALSE with FALSE for the last segment |
| T4 | Terminate_Load<br>Load Type : = upload,<br>Load Service Used : = Pull_Segment |

#### 6.2.6.4.2.4 Table des transitions d'états de téléversement

Le Tableau 65 donne des transitions d'états de téléversement.

**Tableau 65 – Diagramme d'états pour les téléversements en mode Pull ("Pull Upload")**

| Transition | État courant | Événement/Action | État suivant |
|---|---|---|---|
| 1 | UPLOADABLE | Initiate_Load.Ind<br>&&Load Region State of this object is in state DOWNLOADABLE II LOADED II IN USE<br><br>Initiate_Load.Rsp(+)<br>the load region image is segmented in accordance with fixed-length attribute constraints into n segments<br>Segments Left To Transfer : = n -- (local counter) | UPLOADING |
| 2 | UPLOADING | Pull_Segment.Ind<br>&& Segments Left To Transfer > 1<br>&&service succeeds(e.g. compliance with Intersegment Req Time Out attribute constraint if supported)<br><br>Pull_Segment.Rsp(+)<br>Segment Number : = Segments Left To Transfer (if any)<br>with More Follows : = TRUE<br>Segments Left To Transfer : = -1 | UPLOADING |
| 3<br>Exist if supports: Intersegmnt Req Time Out | UPLOADING | Pull_Segment.Ind<br>&&Segments Left To Transfer = any value<br>&&service fails(e.g. no compliance with Intersegment Req Time Out attribute constraint if supported)<br><br>Pull_Segment.Rsp(-) | UPLOADING |
| 4 | UPLOADING | Pull_Segment.Ind<br>&&Segments Left To Transfer=1 (last segment)<br>&&service succeeds(e.g. compliance with Intersegment Req Time Out attribute constraint)<br><br>Pull_Segment.Rsp(+)<br>Segment Number : = Segments Left To Transfer (if any)<br>More Follows : = FALSE<br>Segments Left To Transfer : = -1 | COMPLETING UPLOAD |
| 5 | UPLOADING | Terminate_Upload.Ind<br><br>Terminate_Upload.Rsp(-)<br>Error Info : = object state conflict | UPLOADABLE |
| 6 | UPLOADING or COMPLETING UPLOAD | Abort.Ind | UPLOADABLE |
| 7 | COMPLETING UPLOAD | Terminate_Upload.Ind<br><br>Terminate_Upload.Rsp(+) | UPLOADABLE |
| 8 | UPLOADABLE | Initiate_Load.Ind<br>&& Load Region State attribute is not in state: DOWNLOADABLE II LOADED II IN USE<br><br>MAInitiate_Load.Rsp(-)<br>with Error Info : = object constraint conflict | UPLOADABLE |

#### 6.2.6.4.3 Diagramme d'états de téléchargement en mode Pull

##### 6.2.6.4.3.1 Description

Cette sorte de transfert utilise les services suivants: Initiate Load, Pull Segment et Terminate Load.

Un serveur peut lancer un transfert de téléchargement vers l'aval. Dans ce cas, il émet une primitive de service Initiate_Load.Req vers un client. Cette primitive dit au client de lancer le transfert de téléchargement vers l'aval pour un objet Load Region spécifié. Une fois que le serveur a tiré le contenu, le client émet une primitive Initiate_Load.Rsp(+/-) vers le serveur.

Un client peut lancer les transferts de téléchargement vers l'aval. Dans ce cas, il émet une primitive Initiate_Load.req vers un serveur. Cette primitive dit au serveur de tirer le contenu d'un objet Load Region spécifié. Une fois que le serveur a téléchargé le contenu, il termine le transfert en émettant une primitive Terminate_Load.Req vers le client.

#### 6.2.6.4.3.2 Ordonnancement des primitives de service

Le Tableau 66 résume l'ordonnancement des primitives de service.

**Tableau 66 – Ordonnancement de primitives de service de "Pull download" (téléchargement vers l'aval en mode Pull)**

| Utilisateur de la FAL (un Client) Propriétaire d'un objet appelant | | Utilisateur de la FAL (un Serveur) Propriétaire d'un objet appelé | |
|---|---|---|---|
| Initiate_Load.Ind | ←----(T 1)------ | Initiate_Load.Req | |
| Initiate_Load.Req | -------(T2)-----→ | Initiate_Load.Ind | (Transfert lancé) |
| Initiate_Load.Cnf | ←----(T2)------- | Initiate_Load.Rsp | |
| Pull_Segment.Ind | <-------(T3)----- | Pull_Segment.Req | (Transfert poursuivi) |
| Pull_Segment.Rsp | -----(T3)-----→ | Pull_Segment.Cnf | |
| | | | |
| Terminate_Load.Ind | <-------(T4)----- | Terminate_Load.Req | (Transfert terminé) |
| Terminate_Load.Rsp | ----(T4)-----→ | Terminate_Load.Cnf | |
| Initiate_Load.Rsp | -------(T1)-----→ | Initiate_Load.Cnf | |

#### 6.2.6.4.3.3 Contraintes sur les paramètres de service

Le Tableau 67 résume les contraintes sur les paramètres du service.

**Tableau 67 – Contraintes sur les paramètres du service "Pull download" (téléchargement vers l'aval en mode Pull)**

| Numéro de transition | Contraintes sur les paramètres |
|---|---|
| T1 | Initiate_Load:<br>Calling Load Region Key attribute selected,<br>Load Type : = download,<br>Service To Use : = Pull_Segment,<br>Load Service Initiator : = TRUE |
| T2 | Initiate_Load<br>Called Load Region Key Attribute selected,<br>Load Type : = download<br>Service To Use : = Pull_Segment<br>Load Service Initiator : = FALSE |
| T3 | Pull_Segment<br>Load Type : = download,<br>Segment Number : = ordered 1 to n if any<br>More Follows : = TRUE/FALSE with FALSE for the last segment |
| T4 | Terminate_Load<br>Load Type : = download,<br>Load Service Used : = Pull_Segment<br>Terminate Reason contained in the request |

#### 6.2.6.4.3.4 Table des transitions d'états de téléchargement

Le Tableau 68 donne des transitions d'états de téléchargement vers l'aval.

**Tableau 68 – Table d'états de téléchargement vers l'aval en mode Pull (Pull download)**

| Transition | État courant | Événement/Action | État suivant |
|---|---|---|---|
| 1 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload State = UPLOADABLE<br><br>Initiate_Load.Rsp(+)<br>Pull_Segment.Req | DOWNLOADING |
| 2 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload State <>UPLOADABLE<br><br>Initiate_Load.Rsp(-)<br>Error Info : = object constraint conflict | DOWNLOADABLE |
| 3 | DOWNLOADING | Pull_Segment.Cnf(+)<br>&&More Follows=TRUE<br><br>Store received data<br>Pull_Segment.Req | DOWNLOADING |
| 4 | DOWNLOADING | Pull_Segment.Cnf(+)<br>&&More Follows=FALSE<br><br>Store received data<br>Terminate_Load.Req<br>Terminate Reason : = success | DOWNLOAD SUCCESS |
| 5 | DOWNLOADING | Pull_Segment.Cnf(-)<br><br>Discard received data<br>Terminate_Load.Req<br>Terminate Reason : = failure | DOWNLOAD FAILURE |
| 6 | DOWNLOADING,<br>DOWNLOAD FAILURE,<br>DOWNLOAD SUCCESS | Abort.Ind<br><br>Discard received data | DOWNLOADABLE |
| 7 | DOWNLOAD FAILURE | Terminate_Load.Cnf(+/-)<br><br>Discard received data | DOWNLOADABLE |
| 8 | DOWNLOAD SUCCESS | Terminate_Load.Cnf(-)<br><br>Discard received data | DOWNLOADABLE |
| 9 | DOWNLOAD SUCCESS | Terminate_Load.Cnf(+)<br><br>Update Contents Size attribute if present | LOADED |
| 10 | LOADED | Initiate_Load.Ind<br>&&Upload state attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Initiate_Load.Rsp(+) | DOWNLOADING |
| 11 | LOADED | Discard.Ind<br>&&Upload State attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Discard.Rsp(+) | DOWNLOADABLE |
| 12 | IN USE | Number of Related object In Use attribute incremented<br>&&its new value > 1<br><br>Set Related Object In Use flag value to TRUE if present (Start/Resume) | IN USE |
| 13 | IN USE | Number of Related Object In Use attribute decremented<br>&&its new value >0<br><br>Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill) | IN USE |

| Transition | État courant | Événement/Action | État suivant |
|---|---|---|---|
| 14 | LOADED | Initiate_Load.Ind<br>&&Upload State attribute in state <> UPLOADABLE<br>II Service fails(e.g. non erasable load region contents)<br><br>Initiate_Load.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 15 | LOADED | Discard.Ind<br>&& Upload State attribute in state <> UPLOADABLE<br>II Service fails(e.g. non erasable load region contents)<br><br>Discard.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 16 | LOADED | Number of Related Object In Use attribute incremented<br>&&its new value=1<br><br>Set Related Object In Use flag value to TRUE if any<br>(Start/Stop, Related objects are FI) | IN USE |
| 17 | IN USE | Number of Related Object In Use Attribute decremented<br>&&its new value=0<br><br>Set Related Object In Use flag value to FALSE if any<br>(Reset/Stop/Kill, Related objects are FI) | LOADED |

#### 6.2.6.4.4 Diagramme d'états de téléchargement vers l'aval en mode Push

##### 6.2.6.4.4.1 Description

Cette sorte de transfert utilise les services suivants: Initiate Load, Pull Segment et Terminate Load

Un serveur peut lancer un transfert de téléchargement vers l'aval. Dans ce cas, il émet une primitive de service Initiate_Load.Req vers un client. Cette primitive dit au client de lancer le transfert de téléchargement vers l'aval pour un objet Load Region spécifié. Une fois que le client a poussé le contenu, il émet une primitive Initiate_Load.Rsp (+/-) vers le serveur.

Un client peut lancer un transfert de téléchargement vers l'aval. Dans ce cas, il émet une primitive de service Initiate_Load.Req vers un serveur. Cette primitive dit au serveur que le contenu d'un objet Load Region spécifié sera poussé. Une fois que le serveur a poussé le contenu, il termine le transfert en émettant une primitive Terminate_Load.Req vers le serveur.

##### 6.2.6.4.4.2 Ordonnancement des primitives de service

Le Tableau 69 résume l'ordonnancement des primitives de service.

**Tableau 69 – Ordonnancement de primitives du service "Push download" (téléchargement vers l'aval en mode Push)**

| Utilisateur de la FAL (un Client) Propriétaire d'un objet appelant | | Utilisateur de la FAL (un Serveur) Propriétaire d'un objet appelé | |
|---|---|---|---|
| Initiate_Load.Ind | ←----(T 1)------ | Initiate_Load.Req | |
| Initiate_Load.Req | -------(T2)-----→ | Initiate_Load.Ind | (Transfert lancé) |
| Initiate_Load.Cnf | ←----(T2)------- | Initiate_Load.Rsp | |
| Push_Segment.Req | -------(T3)-----→ | Push_Segment.Ind | (Transfert poursuivi) |
| Push_Segment.Cnf | ←----(T3)------- | Push_Segment.Rsp | |
| | | | |
| Terminate_Load.Req | -------(T4)-----→ | Terminate_Load.Ind | (Transfert terminé) |
| Terminate_Load.Cnf | ←----(T4)------- | Terminate_Load.Rsp | |
| Initiate_Load.Rsp | -------(T1)-----→ | Initiate_Load.Cnf | |

#### 6.2.6.4.4.3 Contraintes sur les paramètres de service

Le Tableau 70 résume les contraintes sur les paramètres de service.

**Tableau 70 – Contraintes sur les paramètres du service "Push download" (téléchargement vers l'aval en mode Push)**

| Numéro de transition | Contraintes sur les paramètres |
|---|---|
| T1 | Initiate_Load:<br>Calling Load Region Key Attribute selected,<br>Load Type : = download,<br>Service To Use : = Push_Segment,<br>Load Service Initiator : = FALSE |
| T2 | Initiate_Load<br>Called Load Region Key Attribute selected,<br>Load Type : = upload<br>Service To Use : = Push_Segment<br>Load Service Initiator : = TRUE |
| T3 | Pull_Segment<br>Load Type : = upload,<br>Segment Number : = ordered 1 to n if any<br>More Follows : = TRUE/FALSE with FALSE for the last segment |
| T4 | Terminate_Load<br>Load Type : = upload,<br>Load Service Used : = Pull_Segment<br>Terminate Reason contained in the response |

#### 6.2.6.4.4.4 Table des transitions d'états de téléchargement vers l'aval

Le Tableau 71 donne des transitions d'états de téléversement.

**Tableau 71 – Table d'états de téléchargement vers l'aval en mode Push (Push download)**

| Transition | État courant | Événement | État suivant |
|---|---|---|---|
| 1 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload state attribute in state UPLOADABLE<br><br>Initiate_Load.Rsp(+) | DOWNLOADING |
| 2 | DOWNLOADABLE | Initiate_Load.Ind<br>&&Upload state attribute in state <>UPLOADABLE<br><br>Initiate_Load.Rsp(-)<br>with Error Info : = object constraint conflict | DOWNLOADABLE |
| 3 | DOWNLOADING | Push_Segment.Ind<br>&&More Follows=TRUE<br>&&compliance with Inter Segment Req Time Out constraint<br><br>Push_Segment.Rsp(+) | DOWNLOADING |
| 4 | DOWNLOADING | Push_Segment.Ind<br>&&More Follows=FALSE<br>&&compliance with Inter Segment Req Time Out constraint<br><br>Push_Segment.Rsp(+) | DOWNLOAD SUCCESS |
| 5<br>exist if supports Inter Segment Req Time Out | DOWNLOADING | Push_Segment.Ind<br>&&More Follows=TRUE II FALSE<br>&&no compliance with Inter Segment Req Time Out constraint<br><br>Push.Segment.Rsp(-)<br>Discard received data | DOWNLOAD FAILURE |
| 6 | DOWNLOADING,<br>DOWNLOAD FAILURE,<br>DOWNLOAD SUCCESS | Abort.Ind<br><br>Discard received data | DOWNLOADABLE |
| 7 | DOWNLOAD FAILURE | Terminate_Load.Ind<br><br>Terminate_Load.Rsp(+/-)<br>Terminate Reason : = success/failure | DOWNLOADABLE |
| 8 | DOWNLOAD SUCCESS | Terminate_Load.Ind<br>&&service fails<br><br>Terminate_Load.Rsp(-)<br>Terminate Reason : = failure<br>Discard received data | DOWNLOADABLE |
| 9 | DOWNLOAD SUCCESS | Terminate_Load.Ind<br>&&service succeeds(e.g. all segments received)<br><br>Update Contents Size attribute if present<br>Terminate_Load.Rsp(+)<br>Terminate Reason : = success | LOADED |
| 10 | LOADED | Initiate_Load.Ind<br>&&Upload State attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Initiate_Load.Rsp(+) | DOWNLOADING |
| 11 | LOADED | Discard.Ind<br>&&Upload State attribute in state UPLOADABLE<br>&&service succeeds(e.g. erasable load region contents)<br><br>Discard.Rsp(+) | DOWNLOADABLE |
| 12 | IN USE | Number of Related object In Use attribute incremented<br>&&its new value > 1 | IN USE |

| Transition | État courant | Événement | État suivant |
|---|---|---|---|
| | | Set Related Object In Use flag value to TRUE if any (Start/Resume) | |
| 13 | IN USE | Number of Related Object In Use attribute decremented<br>&&its new value >0<br><br>Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill) | IN USE |
| 14 | LOADED | Initiate_Load.Ind<br>&&Upload State attribute in state <> UPLOADABLE<br>II Service fails(e.g. non erasable load region contents)<br><br>Initiate_Load.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 15 | LOADED | Discard.Ind<br>&&Upload State attribute in state <> UPLOADABLE<br>\|\| Service fails(e.g. non erasable load region contents)<br><br>Discard.Rsp(-)<br>Error Info : = object constraint conflict | LOADED |
| 16 | LOADED | Number of Related Object In Use attribute incremented<br>&&its new value=1<br><br>Set Related Object In Use flag value to TRUE if any (Start/Stop, Related objects are FI) | IN USE |
| 17 | IN USE | Number of Related Object In Use Attribute decremented<br>&&its new value=0<br><br>Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill, Related objects are FI) | LOADED |

### 6.2.7 ASE Function Invocation

#### 6.2.7.1 Vue d'ensemble

Le modèle d'invocation de fonction de la FAL définit deux objets, à savoir l'objet Action sans état et l'objet Function Invocation orienté état.

Les invocations de fonction sans état, appelées "actions", se déroulent jusqu'à leur achèvement lorsqu'elles sont invoquées et ne peuvent pas être interrompues. En outre, certaines actions atomiques retournent une valeur en réponse à leur invocation, alors que d'autres ne le font pas. Celles qui le font peuvent être utilisées pour modéliser des *fonctions* logicielles, et celles qui ne le font pas peuvent être utilisées pour modéliser des *procédures* logicielles.

D'autre part, les invocations de fonction orientées état peuvent être commandées pendant leur exécution. Des services sont définis pour les suspendre, les reprendre ou les abandonner une fois qu'elles ont été invoquées. Elles ne retournent pas de valeur en réponse à leur lancement. S'il est nécessaire d'abandonner une invocation de fonction une fois qu'elle a démarré, il convient qu'elle soit définie comme étant orientée état.

Les invocations de fonction orientées état représentent la vue réseau d'une invocation spécifique d'une fonction utilisateur. Si la fonction utilisateur peut être invoquée plus d'une fois simultanément, des objets Function Invocation distincts sont nécessaires pour représenter chaque invocation.

Les invocations de fonction orientées état peuvent être utilisées pour modéliser des processus logiciels ou des opérations utilisateur qui peuvent être lancé(e)s et commandé(e)s.

NOTE Un exemple d'invocation de fonction orientée état qui peut être utilisée pour modéliser une opération utilisateur est l'opération "playback" (restitution) d'un enregistreur de vidéos. Une fois qu'elle a été lancée, l'opération de restitution peut être arrêtée (suspendue) et reprise plus tard.

Pour prendre en charge le concept de modélisation de processus logiciel, la définition d'invocations de fonction orientées état inclut des attributs facultatifs qui peuvent être utilisés pour les relier à des objets Load Region.

Le reste du présent paragraphe spécifie les définitions de classe et les services pour l'objet *function invocation* orienté état et l'objet *action* sans état.

#### 6.2.7.2 Spécifications de la classe-modèle Function invocation

##### 6.2.7.2.1 Spécification de la classe Function invocation

###### 6.2.7.2.1.1 Vue d'ensemble de la classe

La classe des invocations de fonctions modélise l'invocation de fonction orientée état. Elle peut être utilisée pour modéliser des processus logiciels ou des fonctions d'utilisateur dont le fonctionnement peut être maîtrisé.

###### 6.2.7.2.1.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE FUNCTION INVOCATION** |
| **CLASS:** | FUNCTION INVOCATION | | |
| **CLASS ID:** | | | 3 |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut: | Access Privilege |
| 1.1 | (m) | Attribut: | Password |
| 1.2 | (m) | Attribut: | Access Groups |
| 1.3 | (m) | Attribut: | Access Rights |
| 2 | (m) | Attribut: | Deletable (TRUE,FALSE) |
| 1 | (m) | Attribut: | Reusable (TRUE,FALSE) Valeur par défaut : TRUE (vrai) |
| 2 | (m) | Attribut: | Function Invocation State |
| 4 | (m) | Attribut: | Number of Related Objects In Use |
| 3 | (o) | Attribut: | List of Related Objects |
| 3.1 | (o) | Attribut: | ClassID |
| 3.2 | (o) | Attribut: | Numeric ID |
| 5.2 | (m) | Attribut: | In Use Flag (TRUE, FALSE) |
| 4 | (c) | Contrainte: | Start Service Supported |
| 4.1 | (o) | Attribut: | Start Service Argument Data type |
| 5 | (c) | Contrainte: | Stop Service Supported |
| 5.1 | (o) | Attribut: | Stop Service Argument Data type |
| 6 | (c) | Contrainte: | Resume Service Supported |
| 6.1 | (o) | Attribut: | Resume Service Argument Data type |
| 7 | (c) | Contrainte: | Reset Service Supported |
| 7.1 | (o) | Attribut: | Reset Service Argument Data type |
| 8 | (c) | Contrainte: | Kill Service Supported |
| 8.1 | (o) | Attribut: | Kill Service Argument Data type |
| 9 | (o) | Attribut: | Last Execution Argument |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Start |
| 2 | (o) | OpsService: | Stop |
| 3 | (o) | OpsService: | Resume |
| 4 | (o) | OpsService: | Reset (réinitialisation) |

5 (o) OpsService: Kill

#### 6.2.7.2.1.3 Attributs

**Access Privilege**

Cet attribut spécifie les contrôles d'accès définis pour cette invocation de fonction. Il se compose des éléments suivants:

**Password**

Cet attribut spécifie le mot de passe pour les droits d'accès. Sa valeur est null s'il n'est pas utilisé.

**Access Groups**

Cet attribut identifie lesquels des huit groupes d'accès définis par un utilisateur sont définis pour l'invocation de fonction. Plus d'un groupe d'accès peut être défini.

**Access Rights**

Cet attribut définit le type d'accès défini pour l'invocation de fonction. Les valeurs valides sont:

- Droit de supprimer les groupes d'accès (voir la NOTE)
- Droit de démarrer les groupes d'accès
- Droit d'arrêter les groupes d'accès
- Droit de reprendre les groupes d'accès
- Droit de réinitialiser les groupes d'accès
- Droit de tuer les groupes d'accès
- Droit de supprimer le mot de passe enregistré (voir NOTE)
- Droit de lancer le mot de passe enregistré
- Droit d'arrêter le mot de passe enregistré
- Droit de reprendre le mot de passe enregistré
- Droit de réinitialiser le mot de passe enregistré
- Droit de tuer le mot de passe enregistré
- Droit de supprimer tous les partenaires de communication (voir NOTE)
- Droit de démarrer tous les partenaires de communication
- Droit d'arrêter tous les partenaires de communication
- Droit de reprendre tous les partenaires de communication
- Droit de réinitialiser tous les partenaires de communication
- Droit de tuer tous les partenaires de communication

NOTE Le droit de supprimer n'a pas de sens si l'objet Function Invocation n'est pas "deletable" (destructible). Ce droit exige l'utilisation du service Object Management Delete, et non un service opérationnel de la classe Function Invocation.

**Deletable**

Cet attribut indique, lorsqu'il est TRUE, que l'invocation de fonction peut être effacée au moyen du service Delete. La valeur de cet attribut est toujours TRUE pour les objets créés dynamiquement avec le service Create de l'ASE Object Management.

**Reusable**

Cet attribut indique, lorsqu'il est TRUE, que l'invocation de fonction peut être exécutée plus d'une fois.

**Function Invocation State**

Cet attribut indique l'état courant de l'invocation de fonction. Un ensemble énuméré de valeurs a été défini et peut être étendu pour une utilisation locale dans l'AP pour décrire des états transitoires de l'invocation de fonction. Ces valeurs d'états transitoires pour cet attribut ne sont toutefois pas visibles sur le réseau. Les transitions entre états et les événements qui provoquent les transitions sont définis par l'utilisateur.

UNRUNNABLE Cet état indique que l'invocation de fonction n'est pas en cours d'exécution et ne peut pas être exécutée.

IDLE Cet état indique que l'invocation de fonction n'est pas en cours d'exécution, mais est capable d'être exécutée.

RUNNING Cet état indique que l'invocation de fonction est en cours d'exécution.

STOPPED Cet état indique que l'exécution d'une invocation de fonction a été suspendue.

STARTING Cet état transitoire indique que l'invocation de fonction est en cours de préparation pour être exécutée. Les invocations de fonction dans cet état sont "ready to run" (prêtes à s'exécuter).

STOPPING Cet état transitoire indique que l'exécution d'une invocation de fonction est en train d'être arrêtée et son contexte sauvegardé.

RESUMING Cet état transitoire indique que l'invocation de fonction est en train d'être retournée à l'état d'exécution à partir d'un contexte sauvegardé précédemment.

RESETTING Cet état transitoire indique que l'invocation de fonction est en train d'être réinitialisée à son contexte initial et d'être retirée de l'exécution.

**Number of Related Objects in Use**

Cet attribut indique le nombre d'objets Load Region connexes définis dans cette invocation de fonction qui sont actuellement dans l'état IN USE. Cet attribut est incrémenté/décrémenté chaque fois qu'un attribut d'état d'objet Load Region connexe entre dans l'état IN USE ou le quitte. Le diagramme d'états de Load Region donne les contraintes relatives à ces transitions.

**List of Related Objects**

Cet attribut facultatif spécifie les objets Load Region connexes à cette invocation de function.

**Class ID**

Cet attribut est l'identificateur de classe de l'objet connexe.

**Numeric ID**

Cet attribut est l'identificateur numérique de l'objet connexe.

**In-use flag**

Cet attribut indique, lorsqu'il est TRUE, que l'objet connexe est actuellement dans l'état IN USE. Cet attribut est mis à jour chaque fois que l'attribut d'état de l'objet Load Region connexe entre dans l'état IN USE ou le quitte. Le diagramme d'états de Load Region donne les contraintes relatives à ces transitions.

**Start Service Argument Data type**

Cet attribut identifie le data type pour chacun des arguments d'exécution pour le service de démarrage (Start) pris en charge par cette invocation de fonction. Cet attribut est présent si l'invocation de fonction prend en charge le service Start (démarrage). Si le service Start est pris en charge, mais n'a pas d'argument d'exécution, la valeur de cet attribut est NULL.

**Stop Service Argument Data type**

Cet attribut identifie le data type pour l'argument d'exécution pour le service d'arrêt (Stop) pris en charge par cette invocation de fonction. Cet attribut est présent si l'invocation de fonction prend en charge le service Stop ('arrêt)Stop). Si le service Stop est pris en charge, mais n'a pas d'argument d'exécution, la valeur de cet attribut est NULL.

**Resume Service Argument Data type**

Cet attribut identifie le data type pour l'argument d'exécution pour le service de reprise (Resume) pris en charge par cette invocation de fonction. Cet attribut est présent si l'invocation de fonction prend en charge le service Resume (reprise). Si le service Resume est pris en charge, mais n'a pas d'argument d'exécution, la valeur de cet attribut est NULL.

**Reset Service Argument Data type**

Cet attribut identifie le data type pour l'argument d'exécution pour le service de réinitialisation (Reset) pris en charge par cette invocation de fonction. Cet attribut est présent si l'invocation de fonction prend en charge le service Reset (réinitialisation). Si le service Reset est pris en charge, mais n'a pas d'argument d'exécution, la valeur de cet attribut est NULL.

**Kill Service Argument Data type**

Cet attribut identifie le data type pour l'argument d'exécution pour le service Kill (tuer) pris en charge par cette invocation de fonction. Cet attribut est présent si l'invocation de fonction prend en charge le service Kill (tuer). Si le service Kill est pris en charge, mais n'a pas d'argument d'exécution, la valeur de cet attribut est NULL.

**Last Execution Argument**

Cet attribut facultatif spécifie la valeur pour le paramètre Execution Argument ('argument d'exécution) qui a été envoyé le dernier par le client.

#### 6.2.7.2.1.4 Services

**Start**

Ce service facultatif est utilisé pour demander à l'invocation de fonction de commencer à s'exécuter. Si ce service n'est pas présent, il est supposé que l'invocation de fonction sera lancée par des moyens locaux.

**Stop**

Ce service facultatif est utilisé pour demander à l'invocation de fonction d'arrêter de s'exécuter.

**Resume**

Ce service facultatif est utilisé pour demander à l'invocation de fonction de continuer son exécution.

**Reset**

Ce service facultatif est utilisé pour demander à l'invocation de fonction de retourner à ses contexte et état initiaux.

**Kill**

Ce service facultatif est utilisé pour demander que l'exécution courante de l'invocation de fonction soit abandonnée et, de ce fait, elle ne peut pas reprendre. Une fois que celle-ci est tuée, le contexte de l'exécution de la fonction est perdu.

### 6.2.7.2.2 Spécification de la classe Action

#### 6.2.7.2.2.1 Vue d'ensemble de la classe

La classe Action est utilisée pour définir des invocations de fonction sans états. Les invocations de fonction sans états s'exécutent jusqu'à la fin lorsqu'elles sont invoquées. Leur exécution ne peut pas être abandonnée en utilisant les services de la FAL. Elles peuvent retourner un résultat en utilisant le service Return.

#### 6.2.7.2.2.2 Modèle formel

**FAL ASE:** **ASE FUNCTION INVOCATION**

| | | | |
|---|---|---|---|
| **CLASS:** | ACTION | | |
| **CLASS ID:** | | 19 | |
| **PARENT CLASS:** | | TOP | |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut: | Reusable (TRUE,FALSE) Valeur par défaut : TRUE (vrai) |
| 2 | (o) | Attribut: | List of Related Objects |
| 3.1 | (m) | Attribut: | ClassID |
| 3.2 | (m) | Attribut: | Numeric ID |
| 3.2 | (m) | Attribut: | In Use Flag (TRUE, FALSE) |
| 4 | (o) | Attribut: | Action Invoke Service Argument Data type |
| 5 | (c) | Contrainte: | Action Return Service Supported |
| 5.1 | (o) | Attribut: | Action Return Service Argument Data type |
| **SERVICES:** | | | |
| 1 | (m) | OpsService: | Action Invoke |
| 2 | (o) | OpsService: | Action Return |

#### 6.2.7.2.2.3 Attributs

**Reusable**

Cet attribut indique, lorsqu'il est TRUE, que l'action peut être exécutée plus d'une fois.

**List of Related Objects**

Cet attribut facultatif spécifie les objets Load Region connexes à cette invocation de fonction.

**Class ID**

Cet attribut est l'identificateur de classe de l'objet connexe.

**Numeric ID**

Cet attribut est l'identificateur numérique de l'objet connexe.

**In-use Flag**

Cet attribut indique, lorsqu'il est TRUE, que l'objet connexe est actuellement dans l'état IN USE. Cet attribut est mis à jour chaque fois que l'attribut état d'objet Load Region entre dans l'état IN USE ou le quitte. Le diagramme d'états de Load Region donne les contraintes relatives à ces transitions.

**Action Invoke Service Argument Data type**

Cet attribut identifie le data type pour l'argument d'exécution pour le service Action Invoke pris en charge par cette action. Cet attribut est présent si l'action prend en charge le service Action Invoke (invocation d'action). Si le service Action Invoke est pris en charge, mais n'a pas d'argument d'exécution, la valeur de cet attribut est NULL.

**Action Return Service Argument Data type**

Cet attribut identifie le data type pour l'argument d'exécution pour le service Action Invoke pris en charge par cette action. Cet attribut est présent si l'action prend en charge le service Action Invoke (invocation d'action). Si le service Action Invoke est pris en charge, mais n'a pas d'argument d'exécution, la valeur de cet attribut est NULL.

#### 6.2.7.2.2.4 Services

**Action Invoke**

Ce service obligatoire est utilisé pour appeler l'objet Action avec une liste d'argument.

**Action Return**

Ce service facultatif est utilisé par l'objet Action pour retourner les résultats de son invocation.

#### 6.2.7.3 Spécifications du service-modèle Function invocation

##### 6.2.7.3.1 Services pris en charge

Ce paragraphe contient la définition de services qui sont propres à cet ASE. Les services définis pour cet ASE sont:

Start
Stop
Resume
Reset
Kill
Action Invoke
Action Return

##### 6.2.7.3.2 Service Start

###### 6.2.7.3.2.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour demander qu'une invocation de fonction soit lancée.

###### 6.2.7.3.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 72.

**Tableau 72 – Paramètres du service "Start"**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

Ce paramètre contient les paramètres de l'invocation de service.

**Attribut-clé**

Ce paramètre spécifie l'un des attributs-clés de l'objet Function Invocation.

**Execution Argument**

Ce paramètre facultatif contient l'argument d'exécution. Ce paramètre est présent lorsque son data type est défini dans la liste des data types d'argument de service pour l'objet Function Invocation.

**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.

**Function Invocation State**

Ce paramètre conditionnel spécifie l'état courant de l'invocation de fonction. Il est présent si le code d'erreur indique un conflit d'états d'objet.

#### 6.2.7.3.2.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.7.3.3 Service Stop

#### 6.2.7.3.3.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour arrêter une invocation de fonction en retenant son contexte afin qu'elle puisse être reprise.

#### 6.2.7.3.3.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 73.

**Tableau 73 – Paramètres du service Stop**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de l'objet Function Invocation.

**Execution Argument**

Ce paramètre facultatif contient l'argument d'exécution. Ce paramètre est présent lorsque son data type est défini dans la liste des data types d'argument de service pour l'objet Function Invocation.

**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.

**Function Invocation State**

Ce paramètre conditionnel spécifie l'état courant de l'invocation de fonction. Il est présent si le code d'erreur indique un conflit d'états d'objet.

#### 6.2.7.3.3.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.7.3.4 Service Resume

#### 6.2.7.3.4.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour reprendre l'exécution d'une invocation de fonction qui a été arrêtée. L'exécution est reprise en utilisant le contexte sauvegardé lorsque l'invocation de fonction avait été arrêtée.

#### 6.2.7.3.4.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 74.

**Tableau 74 – Paramètres du service Resume**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de l'objet Function Invocation.

**Execution Argument**

Ce paramètre facultatif contient l'argument d'exécution. Ce paramètre est présent lorsque son data type est défini dans la liste des data types d'argument de service pour l'objet Function Invocation.

**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.

**Function Invocation State**

Ce paramètre conditionnel spécifie l'état courant de l'invocation de fonction. Il est présent si le code d'erreur indique un conflit d'états d'objet.

#### 6.2.7.3.4.3 Procédure de service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.7.3.5 Service Reset

#### 6.2.7.3.5.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour réinitialiser une invocation de fonction avec son contexte initial. Son contexte initial est défini comme étant le contexte de l'invocation de fonction établi par ses procédures d'initialisation.

#### 6.2.7.3.5.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 75.

**Tableau 75 – Paramètres du service "Reset"**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de l'objet Function Invocation.

**Execution Argument**

Ce paramètre facultatif contient l'argument d'exécution. Ce paramètre est présent lorsque son data type est défini dans la liste des data types d'argument de service pour l'objet Function Invocation.

**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.

**Function Invocation State**

Ce paramètre conditionnel spécifie l'état courant de l'invocation de fonction. Il est présent si le code d'erreur indique un conflit d'états d'objet.

#### 6.2.7.3.5.3 Procédure du service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service.

### 6.2.7.3.6 Service Kill

#### 6.2.7.3.6.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour abandonner l'exécution d'une invocation de fonction afin qu'elle ne puisse pas être redémarrée ou reprise.

#### 6.2.7.3.6.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 76.

**Tableau 76 – Paramètres du service Kill**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Key Attribute | M | M (=) | | |
| Execution Argument | U | U (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Error Info | | | M | M (=) |
| Function Invocation State | | | C | C (=) |
| NOTE Voir la Note en 3.8.4.3. | | | | |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de l'objet Function Invocation.

**Execution Argument**

Ce paramètre facultatif contient l'argument d'exécution. Ce paramètre est présent lorsque son data type est défini dans la liste des data types d'argument de service pour l'objet Function Invocation.

**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.

**Function Invocation State**

Ce paramètre conditionnel spécifie l'état courant de l'invocation de fonction. Il est présent si le code d'erreur indique un conflit d'états d'objet.

#### 6.2.7.3.6.3 Procédure du service

La procédure de service confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.7.3.7 Service Action invoke

#### 6.2.7.3.7.1 Vue d'ensemble du service

Ce service non confirmé est utilisé pour invoquer l'exécution d'un objet Action.

#### 6.2.7.3.7.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 77.

**Tableau 77 – Paramètres du service Action invoke**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Key Attribute | M | M (=) |
| Action Invocation ID | M | M (=) |
| Execution Argument | C | C (=) |
| Timeliness | | C |
| Duplicate FAL PDU Body | | C |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Destination DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse de destination à laquelle l'invocation d'action demandée est à envoyer.

**Source DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse source à partir de laquelle l'invocation d'action indiquée est à envoyer.

**Key Attribute**

Ce paramètre spécifie l'un des attributs-clés de l'objet Action.

**Action Invocation ID**

Ce paramètre identifie une invocation spécifique de l'objet Action. Il permet la corrélation de l'invocation et de son résultat tel que rapporté par un service Return.

**Execution Argument**

Ce paramètre facultatif contient l'argument d'exécution. Ce paramètre est présent lorsque son data type est défini dans la liste des data types d'argument de service pour l'objet Action.

**Timeliness**

Ce paramètre conditionnel indique le statut de cohérence temporelle de couche liaison de données pour le transfert de cette demande de service. Ce paramètre est présent s'il est pris en charge sur l'AR spécifiée.

**Duplicate FAL PDU Body**

Ce paramètre conditionnel indique si, oui ou non, la réception d'une PDU de FAL dupliquée a été détectée par la couche liaison de données. Il est présent s'il est pris en charge dans les attributs de mapping de la DL de l'AREP.

#### 6.2.7.3.7.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

### 6.2.7.3.8 Service Action return

#### 6.2.7.3.8.1 Vue d'ensemble du service

Ce service non confirmé est utilisé pour retourner les résultats d'une invocation d'un objet Action.

#### 6.2.7.3.8.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 78.

**Tableau 78 – Paramètres du service Action return**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| AREP | M | M |
| Destination DL-Address | C | |
| Source DL-Address | | C |
| Action Invocation ID | M | M (=) |
| Action Success | S | S (=) |
| Return Value | U | U (=) |
| Action Failure | S | S (=) |
| Error Info | M | M (=) |

**Argument**

Ce paramètre contient les paramètres de l'invocation du service.

**Destination DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse de destination à laquelle le retour d'action (Action Return) demandé est à envoyer.

**Source DL-Address**

Ce paramètre n'existe que lorsque l'AREP correspondant le prend en charge et est configuré avec un attribut Remote Address Configuration Type mis à FREE. Il indique l'adresse source à partir de laquelle le retour d'action indiqué est à envoyer.

**Action Invocation ID**

Ce paramètre identifie l'invocation spécifique de l'objet Action dont les résultats sont actuellement retournés.

**Action Success**

Ce paramètre de type sélection est présent si l'action a réussi.

**Return Value**

Ce paramètre facultatif contient le résultat de cette invocation de l'action.

**Action Failure**

Cet argument de type sélection est présent si l'action n'a pas réussi.

#### 6.2.7.3.8.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

L'AP contenant l'action présente une primitive "request" du service Action Return pour retourner les résultats d'une précédente invocation de l'action. Un identificateur d'invocation d'action est inclus dans la demande pour identifier l'invocation particulière. Cet identificateur n'est utilisé que par l'utilisateur destinataire et n'est acheminé que par la FAL

### 6.2.7.4 Diagramme d'états de Function Invocation

Le Tableau 79 donne les transitions d'états d'un objet Function Invocation. L'état “NON EXISTENT” indique que l'objet n'existe pas. Il n'est pas défini comme une valeur valide pour l'attribut Function Invocation State, car les attributs sont créés et supprimés avec l'objet (et n'ont donc pas de valeur lorsque l'objet n'existe pas).

**Tableau 79 – Transitions d'états pour un objet Function Invocation**

| Transition | État courant | Events | État suivant |
|---|---|---|---|
| 1 | NON EXISTENT | Create.Ind<br>&& List of Attribute IDs and Initial Values parameter = List of Load Region Object IDs<br>&& Specified Load Region objects available<br><br>Pour l'objet Function Invocation, mettre à jour les ATTRIBUTES:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les ATTRIBUTES:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Create.Rsp(+) | Idle |
| 2 | NON EXISTENT | Create.Ind<br>&& List of Attribute IDs and Initial Values parameter <> List of Load Region Object IDs<br>\|\| Specified Load Region objects not available<br><br>Create.Rsp(-) | NON EXISTENT |
| 3 | IDLE | Delete.Ind<br>&& Deleteable=TRUE<br><br>Delete.Rsp(+) | NON EXISTENT |
| 4 | IDLE | Delete.Ind<br>&& Deleteable=FALSE<br><br>Delete.Rsp(-) | IDLE |
| 5 | IDLE | Start.Ind | STARTING |
| x | STARTING | Start succeeds: par exemple<br>Related Load Region Object in state LOADED or IN-USE<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Start.Rsp(+) | RUNNING |
| 7 | STARTING | Start fails, non destructive: par exemple<br>Related Load Region Object not in state LOADED or IN USE<br><br>Start.Rsp(-) | IDLE |
| 8 | STARTING | Start fails, destructive<br><br>Start.Rsp(-) | UN-RUNNABLE |
| 9 | RUNNING | Stop.Ind | STOPPING |
| 10 | STOPPING | Stop succeeds: par exemple<br>Related Load Region Object in state IN USE<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Stop.Rsp(+) | STOPPED |
| 11 | STOPPING | Stop Execution fails, non destructive<br><br>Stop.Rsp(-) | RUNNING |

| Transition | État courant | Events | État suivant |
|---|---|---|---|
| 12 | STOPPING | Stop fails, destructive<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Stop.Rsp(-) | UN-RUNNABLE |
| 13 | STOPPED | Resume.Ind | RESUMING |
| 14 | RESUMING | Resume succeeds: par exemple<br>Related Load Region Object in state Loaded or In-Use<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Resume.Rsp(+) | RUNNING |
| 15 | RESUMING | Resume fails, non destructive: par exemple<br>Load Region not in state LOADED or IN-USE<br><br>Resume.Rsp(-) | STOPPED |
| 16 | RESUMING | Resume fails, destructive<br><br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Resume.Rsp(-) | UNRUNNABLE |
| 17 | STOPPED | Reset.Ind | RESETTING |
| 18 | RESETTING | Reset succeeds, reusable=TRUE<br><br>Reset.Rsp(+) | IDLE |
| 19 | RESETTING | Reset succeeds, reusable = false<br><br>Reset.Rsp(+) | UNRUNNABLE |
| 20 | RESETTING | Reset fails, non destructive<br><br>Reset.Rsp(-) | STOPPED |
| 21 | RESETTING | Reset fails, destructive<br><br>Reset.Rsp(-) | UNRUNNABLE |
| 22 | RESETTING | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 23 | STOPPING | Kill.Ind<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Kill.Rsp(+) | UNRUNNABLE |

| Transition | État courant | Events | État suivant |
|---|---|---|---|
| 24 | RUNNING | End of Program<br>&& Reusable=TRUE<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported) | IDLE |
| 25 | RUNNING | End of Program<br>&& Reusable = false<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported) | UNRUNNABLE |
| 26 | RUNNING | Program stopped<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported) | STOPPED |
| 27 | STOPPED | Delete.Ind<br>&& this Function Invocation is not deletable<br><br>Delete.Rsp(-) | STOPPED |
| 28 | STOPPED | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 29 | STOPPED | Delete.Ind<br>&& this Function Invocation is deletable<br><br>Delete.Rsp(+) | NON EXISTENT |
| 30 | RESUMING | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 31 | RUNNING | Kill.Ind<br><br>Pour l'objet Function Invocation, mettre à jour les attributs:<br>Number of Related Object In Use<br>List of Related Objects( if supported)<br>Pour chaque objet Load Region connexe, mettre à jour les attributs:<br>Number of Related Objects In Use<br>List of Related Objects( if supported)<br>Kill.Rsp(+) | UNRUNNABLE |
| 32 | STARTING | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 33 | IDLE | Kill.Ind<br><br>Kill.Rsp(+) | UNRUNNABLE |
| 34 | UNRUNNABLE | Delete.Ind<br><br>Delete.Rsp(+) | NON EXISTENT |

## 6.3 AR

### 6.3.1 Spécification de la classe Queued user-triggered unidirectional (QUU) AR endpoint

#### 6.3.1.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge la distribution en file d'attente de la demande de services non confirmés vers un ou plusieurs processus application. Le comportement de ce type d'AR peut être décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande présente une unité de données de service d'ASE AR au point d'extrémité expéditeur de l'AR. L'AREP qui envoie l'APDU de demande la présente à sa couche sous-jacente en vue de son transfert. La couche sous-jacente l'envoie à sa prochaine occasion. L'AREP qui reçoit l'APDU de demande provenant de sa couche sous-jacente la livre à l'utilisateur d'ASE d'AR dans l'ordre dans lequel elle a été reçue.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | REPORT SOURCE<br>REPORT SINK |
| Cardinalité: | 1 à n |
| Cohérence temporelle: | Non |

#### 6.3.1.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | Queued User-triggered Uni-directional AR Endpoint | | |
| **CLASS ID:** | | | 36 |
| **PARENT CLASS:** | | | AR Endpoint |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1. | (m) | Attribut: | Role (REPORT SOURCE, REPORT SINK) |
| 2. | (m) | Attribut: | AREP State (OPEN, CLOSED) |
| 3. | (m) | Attribut: | Remote Address Configuration Type (FREE, LINKED) |
| 4. | (m) | Attribut: | DL Mapping Reference |
| **SERVICES:** | | | |
| 1. | (o) | OpsService: | Unconfirmed Send (Envoi de services non confirmés) |

#### 6.3.1.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

REPORT SOURCE Des points d'extrémité de ce type distribuent des rapports vers plusieurs points d'extrémité en utilisant des services non confirmés opérant sur des services de liaison de données sans connexion.

REPORT SINK Des points d'extrémité de ce type reçoivent des rapports issus de points d'extrémité source en utilisant des services non confirmés opérant sur des services de liaison de données sans connexion.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Remote Address Configuration Type**

Cet attribut spécifie comment l'adresse distante de l'AREP est configurée.

FREE La valeur “FREE” indique que la destination des PDU de la FAL issues de l'AREP source est fournie dynamiquement dans la demande de service non confirmé.

LINKED La valeur “LINKED” indique que la destination des PDU de la FAL est configurée avec l'attribut RemoteDlsapAddress contenu dans le Mapping de DL (spécifié dans la CEI 61158-6-5).

**DL Mapping Reference**

Cet attribut est une référence au mapping de la couche liaison de données sous-jacente. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

#### 6.3.1.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé.

### 6.3.2 Spécification de la classe Queued user-triggered bi-directional connection-oriented (QUB-Co) AR endpoint

#### 6.3.2.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge l'échange, à la demande, de services confirmés entre deux processus application. Les services non confirmés ne sont pas pris en charge par ce type d'AR. Elle utilise des services de liaison de données en mode orienté connexion pour les échanges. Le comportement de cette classe est décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande la présente comme une unité de données de service d'ASE d’AR à son AREP. L'AREP qui envoie l'APDU de demande la met en file d'attente à la couche sous-jacente en vue de son transfert à la prochaine occasion.

L'AREP qui reçoit l'APDU de demande provenant de sa couche sous-jacente la met en file d'attente en vue de sa distribution à son utilisateur d'ASE AR, dans l'ordre dans lequel elle a été reçue.

Pour une demande de services confirmés, l'AREP qui reçoit l'APDU de demande accepte l'APDU de réponse correspondante provenant de son utilisateur d'ASE AR et la met en file d'attente à la couche sous-jacente en vue de son transfert.

L'AREP qui a émis l'APDU de demande reçoit l'APDU de réponse provenant de sa couche sous-jacente et la met en file d'attente en vue de sa distribution à son utilisateur d'ASE AR dans l'ordre dans lequel elle a été reçue. Il arrête également son temporisateur associé de réponse de service.

Les caractéristiques de cette classe AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | Client<br>Server<br>Peer (homologue/pair) |
| Cardinalité: | 1 à 1 |
| Cohérence temporelle: | Non |

#### 6.3.2.2 Modèle formel

| | |
|---|---|
| **FAL ASE:** | **ASE AR** |
| **CLASS:** | Queued User-triggered Bi-directional Connection-Oriented AREP |
| **CLASS ID:** | 34 |

**PARENT CLASS:** AR Endpoint

**ATTRIBUTS DE GESTION DE RÉSEAU**

| | | | |
|---|---|---|---|
| 1. | (m) | Attribut: | Role (CLIENT, SERVER, PEER) |
| 2. | (m) | Attribut: | AREP State |
| 3. | (m) | Attribut: | Remote Address Configuration Type (FREE, LINKED) |
| 4. | (m) | Attribut: | Maximum Outstanding Requests Calling |
| 5. | (m) | Attribut: | Maximum Outstanding Requests Called |
| 6. | (m) | Attribut: | Initiator (TRUE, FALSE) |
| 7. | (o) | Attribut: | Remote AP Name |
| 8. | (m) | Attribut: | Transmit DL Mapping Reference |
| 9. | (m) | Attribut: | Receive DL Mapping Reference |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1. | (o) | OpsService: | Confirmed Send |
| 2. | (o) | OpsService: | Establish |

#### 6.3.2.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PEER (CLIENT/SERVER) Les points d'extrémité de ce type peuvent agir comme clients, comme serveurs ou simultanément comme les deux. Il convient que les points d'extrémité de ce type indiquent si, oui ou non, il faut qu'ils soient les initiateurs du processus d'établissement d'AR.

CLIENT Les points d'extrémité de ce type émettent des APDU de demande de services confirmés vers les serveurs et reçoivent des APDU de réponse de services confirmés.

SERVER Les points d'extrémité de ce type reçoivent des APDU de demande de services confirmés et non confirmés provenant des clients et émettent vers ceux-ci des APDU de réponse de services confirmés.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Remote Address Configuration Type**

Cet attribut spécifie comment l'adresse distante de l'AREP est configurée. Les valeurs valides sont:

FREE La valeur "FREE" indique que la destination des PDU de la FAL issues de l'ARP source est fournie dynamiquement dans la demande de service non confirmé.

LINKED La valeur "LINKED" indique que la destination des PDU de la FAL est configurée avec l'attribut RemoteDlsapAddress contenu dans le Mapping de DL (spécifié dans la CEI 61158-6-5).

**Max Outstanding Requests Calling**

Cet attribut conditionnel indique le nombre maximal de réponses que l'AREP peut attendre de l'AREP distant.

**Max Outstanding Requests Called**

Cet attribut conditionnel indique le nombre maximal de réponses que le point d'extrémité AREP peut attendre de son utilisateur local.

**Initiator**

Cet attribut indique, lorsqu'il est TRUE, que le point d'extrémité a été configuré pour lancer l'établissement de l'AR. Il convient qu'un et un seul des AREP impliqués dans une AR soit l'initiateur. Lorsque cet AREP est un client, la valeur de cet attribut est toujours TRUE. Lorsque cet AREP est un serveur, la valeur de cet attribut est toujours FALSE. Lorsque cet AREP est un Peer (c'est-à-dire un homologue), la valeur de cet attribut peut être l'un ou l'autre, tant que les deux AREP dans l'AR ne sont pas configurés pour être l'initiateur.

**Remote AP Name**

Cet attribut facultatif spécifie le nom de l'AP attaché à l'AREP distant.

**Transmit DL Mapping Reference**

Cet attribut fournit une référence au mapping de la couche liaison de données sous-jacente pour le trajet d'acheminement de l'émission pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**Receive DL Mapping Reference**

Cet attribut fournit une référence au mapping de la couche liaison de données sous-jacente pour le trajet d'acheminement de la réception pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

#### 6.3.2.4 Services

**Confirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service confirmé.

**Establish**

Ce service est utilisé pour établir une AR.

### 6.3.3 Spécification de la classe Queued user-triggered bi-directional connectionless (QUB-CL) AR endpoint

#### 6.3.3.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge l'échange, à la demande, de services confirmés et non confirmés entre deux ou plusieurs processus application. Cette classe utilise des services de liaison de données sans connexion pour les échanges. Les transferts de couche liaison de données peuvent être acquittés ou non acquittés. Le comportement de cette classe est décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande la présente comme une unité de données de service d'ASE AR à son AREP. L'AREP qui envoie l'APDU de demande la met en file d'attente à la couche sous-jacente en vue de son transfert à la prochaine occasion.

L'AREP qui reçoit l'APDU de demande provenant de sa couche sous-jacente la met en file d'attente en vue de sa distribution à son utilisateur d'ASE d'AR dans l'ordre dans lequel elle a été reçue.

Pour une demande de services confirmés, l'AREP qui reçoit l'APDU de demande accepte l'APDU de réponse correspondante provenant de son utilisateur d'ASE AR et la met en file d'attente à la couche sous-jacente en vue de son transfert.

L'AREP qui a émis l'APDU de demande reçoit l'APDU de réponse provenant de sa couche sous-jacente et la met en file d'attente en vue de sa distribution à son utilisateur d'ASE d'AR dans l'ordre dans lequel elle a été reçue. Il arrête également son temporisateur associé de réponse de service.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | Client<br>Server<br>Peer (homologue/pair) |
| Cardinalité: | 1 à n |
| Cohérence temporelle: | Non |

#### 6.3.3.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | Queued User-triggered Bi-directional Connectionless AREP | | |
| **CLASS ID:** | | | 45 |
| **PARENT CLASS:** | | | AR Endpoint |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1 | (m) | Attribut: | Role (CLIENT, SERVER, PEER) |
| 2 | (m) | Attribut: | AREP State |
| 3 | (m) | Attribut: | Remote Address Configuration Type (FREE, LINKED) |
| 4 | (m) | Attribut: | Maximum Outstanding Requests Calling |
| 5 | (m) | Attribut: | Maximum Outstanding Requests Called |
| 6 | (m) | Attribut: | Initiator (TRUE, FALSE) |
| 7 | (o) | Attribut: | Remote AP Name |
| 8 | (m) | Attribut: | Transmit DL Mapping Reference |
| 9 | (m) | Attribut: | Receive DL Mapping Reference |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Confirmed Send |
| 2 | (o) | OpsService: | Unconfirmed Send |
| 3 | (o) | OpsService: | Establish |
| 4 | (o) | OpsService: | Abort |

#### 6.3.3.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PEER (CLIENT/SERVER) Les points d'extrémité de ce type peuvent agir comme clients, comme serveurs ou simultanément comme les deux. Il convient que les points d'extrémité de ce type indiquent si, oui ou non, il faut qu'ils soient les initiateurs du processus d'établissement d'AR.

CLIENT Les points d'extrémité de ce type émettent des APDU de demande de services confirmés et non confirmés vers les serveurs et reçoivent des APDU de réponse de services confirmés.

SERVER Les points d'extrémité de ce type reçoivent des APDU de demande de services confirmés et non confirmés provenant des clients et émettent vers ceux-ci des APDU de réponse de services confirmés. Ils peuvent également émettre des APDU de demande de services non confirmés vers des clients.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Remote Address Configuration Type**

Cet attribut spécifie comment l'adresse distante de l'AREP est configurée. Les valeurs valides sont:

FREE La valeur “FREE” indique que la destination des PDU de la FAL issues de l'ARP source est fournie dynamiquement.

LINKED La valeur “LINKED” indique que la destination des PDU de la FAL est configurée avec l'attribut RemoteDlsapAddress contenu dans le Mapping de DL (spécifié dans la CEI 61158-6-5).

**Max Outstanding Requests Calling**

Cet attribut conditionnel indique le nombre maximal de réponses que l'AREP peut attendre de l'AREP distant.

**Max Outstanding Requests Called**

Cet attribut conditionnel indique le nombre maximal de réponses que le point d'extrémité AREP peut attendre de son utilisateur local.

**initiator** (initiateur)

Cet attribut indique, lorsqu'il est TRUE, que le point d'extrémité a été configuré pour lancer l'établissement de l'AR. Il convient qu'un et un seul des AREP impliqués dans une AR soit l'initiateur. Lorsque cet AREP est un client, la valeur de cet attribut est toujours TRUE. Lorsque cet AREP est un serveur, la valeur de cet attribut est toujours FALSE. Lorsque cet AREP est un Peer (c'est-à-dire un homologue), la valeur de cet attribut peut être l'un ou l'autre, tant que les deux AREP de l'AR ne sont pas configurés pour être l'initiateur.

**Remote AP Name**

Cet attribut facultatif spécifie le nom de l'AP attaché à l'AREP distant.

**Transmit DL Mapping Reference**

Cet attribut fournit une référence au mapping à la couche liaison de données sous-jacente pour le trajet d'acheminement en émission pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**Receive DL Mapping Reference**

Cet attribut fournit une référence au mapping à la couche liaison de données sous-jacente pour le trajet d'acheminement en réception pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

#### 6.3.3.4 Services

**Confirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service confirmé.

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé.

**Establish**

Ce service est utilisé pour établir une AR.

**Abort**

Ce service est utilisé pour abandonner une AR.

### 6.3.4 Spécification de la classe Queued user-triggered bi-directional with flow control (QUB-FC) AR endpoint

#### 6.3.4.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge le fonctionnement asynchrone du modèle client/serveur utilisant des files d'attente. Elle permet l'acheminement de services confirmés et non confirmés en utilisant le contrôle de flux de la couche application pour les services non confirmés et utilise les services de liaison de données orientée connexion pour les échanges.

La priorité de liaison de données pour les transferts est spécifiée séparément pour chaque transfert.

Le comportement de cette classe est décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande la présente comme une unité de données de service d'ASE AR à son AREP. L'AREP qui envoie l'APDU de demande la met en file d'attente à sa couche sous-jacente en vue de son transfert à la prochaine occasion disponible, sauf les services acquittés non confirmés. Cette dernière sera ignorée si le compteur de services en souffrance a atteint le maximum négocié.

L'AREP qui reçoit l'APDU de demande provenant de sa couche sous-jacente la met en file d'attente en vue de sa distribution à son utilisateur d'ASE AR dans l'ordre dans lequel elle a été reçue. Si l'APDU reçu contient une demande acquittée de services non confirmés, l'AREP émet l'indication appropriée vers l'utilisateur d'ASE AR et une UCA_AckPDU vers la couche sous-jacente en vue du transfert.

Pour une demande de services confirmés, l'AREP qui reçoit l'APDU de demande accepte l'APDU de réponse correspondante provenant de son utilisateur d'ASE AR et la met en file d'attente à la couche sous-jacente en vue de son transfert.

L'AREP qui a émis l'APDU de demande reçoit l'APDU de réponse provenant de sa couche sous-jacente et la met en file d'attente en vue de sa distribution à son utilisateur d'ASE AR, dans l'ordre dans lequel elle a été reçue, et décrémente le compteur de contrôle de flux.

En outre, l'AREP surveille la connexion au cas où aucun service issu de l'utilisateur d'ASE AR n'est en souffrance en envoyant une IdlePDU à l'instance homologue. Le TCTimer (Transmit Client Timer, c'est-à-dire Temporisateur client en mode émission) et le TSTimer (Transmit Server Timer, c'est-à-dire Temporisateur serveur en mode émission) surveillent l'envoi des dlePDUs sur les deux nœuds de connexion. Le RCTimer (Receive Client Timer, c'est-à-dire Temporisateur client en mode réception) et le RSTimer (Receive Server Timer, c'est-à-dire Temporisateur serveur en mode réception) surveillent la vie de la connexion sur les deux noeuds de connexion. Le RCTimer ou le RSTimer expire si aucune PDU (et aussi aucune IdlePDU) n'a été reçue pendant une certaine durée et entraîne l'arrêt de la connexion de l'AREP.

De surcroît, l'utilisateur d'ASE AR au niveau du serveur peut interrompre ou poursuivre le flux de données en émettant le service AR-XON-XOFF.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles | Client<br>Serveur<br>Homologue |
| Cardinalité | 1 à 1 |
| Cohérence temporelle | Non |

#### 6.3.4.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | | | Queued User-triggered Bi-directional with Flow Control AREP |
| **CLASS ID:** | | | 46 |
| **PARENT CLASS:** | | | AR Endpoint |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1 | (m) | Attribut: | Role (CLIENT, SERVER, PEER) |
| 2 | (m) | Attribut: | AREP State |
| 3 | (m) | Attribut: | Remote Address Configuration Type (FREE, LINKED) |

| | | | |
|---|---|---|---|
| 4 | (m) | Attribut: | Maximum Outstanding Requests Calling |
| 5 | (m) | Attribut: | Maximum Outstanding Requests Called |
| 6 | (m) | Attribut: | Max Outstanding Unconfirmed Requests Client (maxUCSC) |
| 7 | (m) | Attribut: | Max Outstanding Unconfirmed Requests Server (maxUCSS) |
| 8 | (m) | Attribut: | Maximum Outstanding Requests Client (maxOSCC) |
| 9 | (m) | Attribut: | Maximum Outstanding Requests Server (maxOSCS) |
| 10 | (m) | Attribut: | Initiator (TRUE, FALSE) |
| 11 | (o) | Attribut: | Remote AP Name |
| 12 | (m) | Attribut: | Transmit DL Mapping Reference |
| 13 | (m) | Attribut: | Receive DL Mapping Reference |
| 14 | (m) | Attribut: | CIU |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Confirmed Send |
| 2 | (o) | OpsService: | Unconfirmed Send |
| 3 | (o) | OpsService: | Establish |
| 4 | (o) | OpsService: | Abort |

#### 6.3.4.3 Attributs de gestion réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PEER (CLIENT/SERVER) Les points d'extrémité de ce type peuvent agir comme clients, comme serveurs ou simultanément comme les deux. Il convient que les points d'extrémité de ce type indiquent si, oui ou non, il faut qu'ils soient les initiateurs du processus d'établissement d'AR.

CLIENT Les points d'extrémité de ce type émettent des APDU de demande de services confirmés et non confirmés vers les serveurs et reçoivent des APDU de réponse de services confirmés.

SERVER Les points d'extrémité de ce type reçoivent des APDU de demande de services confirmés et non confirmés provenant des clients et émettent vers ceux-ci des APDU de réponse de services confirmés. Ils peuvent également émettre des APDU de demande de services non confirmés vers des clients.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Remote Address Configuration Type**

Cet attribut spécifie comment l'adresse distante de l'AREP est configurée. Les valeurs valides sont:

FREE La valeur "FREE" indique que la destination des PDU de la FAL issues de l'ARP source est fournie dynamiquement dans la demande de service non confirmé.

LINKED La valeur "LINKED" indique que la destination des PDU de la FAL est configurée avec l'attribut RemoteDlsapAddress contenu dans le Mapping de DL (spécifié dans la CEI 61158-6-5).

**Max outstanding Requests Calling**

Cet attribut indique le nombre maximal de réponses que l'AREP peut attendre de l'AREP distant.

**Max Outstanding Requests Called**

Cet attribut indique le nombre maximal de réponses que l'AR du point d'extrémité peut attendre de son utilisateur local.

**Max Outstanding Unconfirmed Requests Client (maxUCSC)**

Cet attribut indique le nombre maximal de demandes que l'AR du point d'extrémité peut attendre de son utilisateur local.

**Max Outstanding Unconfirmed Requests Server (maxUCSS)**

Cet attribut indique le nombre maximal de demandes que l'AREP peut attendre de l'AREP distant.

**Maximum Outstanding Requests Client (maxOSCC)**

Cet attribut indique le nombre maximal de réponses que l'AREP peut attendre de l'AREP distant.

**Maximum Outstanding Requests Server (maxOSCS)**

Cet attribut indique le nombre maximal de réponses AR que le point d'extrémité peut attendre de son utilisateur local.

**Initiator** (initiateur)

Cet attribut indique, lorsqu'il est TRUE, que le point d'extrémité a été configurée pour lancer l'établissement de l'AR. Il convient qu'un et un seul des AREP impliqués dans une AR soit l'initiateur. Lorsque cet AREP est un client, la valeur de cet attribut est toujours TRUE. Lorsque cet AREP est un serveur, la valeur de cet attribut est toujours FALSE. Lorsque cet AREP est un Peer (c'est-à-dire un homologue), la valeur de cet attribut peut être l'un ou l'autre, tant que les deux AREP dans l'AR ne sont pas configurés pour être l'initiateur.

**Remote AP Name**

Cet attribut facultatif spécifie le nom de l'AP attaché à l'AREP distant.

**Transmit DL Mapping Reference**

Cet attribut fournit une référence au mapping à la couche liaison de données sous-jacente pour le trajet d'acheminement de l'émission pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**Receive DL Mapping Reference**

Cet attribut fournit une référence au mapping à la couche liaison de données sous-jacente pour le trajet d'acheminement de la réception pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**CIU**

Cet attribut (CIU – Control Interval User-triggered, c'est-à-dire Intervalle de commande pour connexion déclenchée par un utilisateur) contient l'intervalle de commande pour surveiller une connexion déclenchée par un utilisateur. Cet attribut est mis par la gestion réseau et est utilisé par l'ARPM. Si cet attribut spécifie une valeur différente de 0, l'ARPM émet une IdlePDU tant que l'ASE AR n'a pas transmis de service pendant l'intervalle de commande (CIU/3). Si CIU expire, il sera mis fin aux connexions.

#### 6.3.4.4 Services

**Confirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service confirmé.

**Establish**

Ce service est utilisé pour établir une AR.

**Abort**

Ce service est utilisé pour déconnecter une AR.

**Unconfirmed Send**

Ce service est utilisé pour envoyer sur une AR un service non confirmé.

### 6.3.5 Spécification de la classe Queued user-triggered bi-directional with segmentation (QUB-Seg) AR

#### 6.3.5.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge l'échange, à la demande, de services confirmés et non confirmés entre deux processus application. Elle utilise des services de liaison de données en mode orienté connexion pour les échanges. Elle prend en charge la segmentation des APDU au sein de la couche application. La priorité de liaison de données pour les transferts est spécifiée séparément pour chaque sens de transfert.

Une unité de données de service d'ASE AR (si sa taille est supérieure à un élément transférable pris séparément) peut être divisée en plusieurs APDU qui sont envoyées vers sa couche sous-jacente pour être transférées. La couche sous-jacente envoie chaque segment à l'occasion suivante. L'AREP qui reçoit les segments d'APDU de demande provenant de sa couche sous-jacente réassemble l'APDU et la livre à l'utilisateur d'ASE AR. La livraison de plusieurs APDU est faite dans l'ordre dans lequel elles sont reçues au complet.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | Client<br>Serveur<br>Homologue |
| Cardinalité: | 1 à 1 |
| Cohérence temporelle: | Non |

#### 6.3.5.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | ASE AR |
| **CLASS:** | | | Queued User-triggered Bi-directional Segmentation AREP |
| **CLASS ID:** | | | 45 |
| **PARENT CLASS:** | | | AR Endpoint |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1. | (m) | Attribut: | Role (CLIENT, SERVER, PEER) |
| 2. | (m) | Attribut: | AREP State |
| 3. | (m) | Attribut: | Remote Address Configuration Type (FREE, LINKED) |
| 4. | (m) | Attribut: | Maximum Outstanding Requests Calling |
| 5. | (m) | Attribut: | Maximum Outstanding Requests Called |
| 6. | (m) | Attribut: | Initiator (TRUE, FALSE) |
| 7. | (o) | Attribut: | Remote AP Name |
| 8. | (m) | Attribut: | Transmit DL Mapping Reference |
| 9. | (m) | Attribut: | Receive DL Mapping Reference |
| **SERVICES:** | | | |
| 1. | (o) | OpsService: | Confirmed Send |
| 2. | (o) | OpsService: | Unconfirmed Send |
| 3. | (o) | OpsService: | Establish |
| 4. | (o) | OpsService: | Reject |
| 5. | (o) | OpsService: | Abort |

#### 6.3.5.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PEER (CLIENT/SERVER) Les points d'extrémité de ce type peuvent agir comme clients, comme serveurs ou simultanément comme les deux. Il convient que les points d'extrémité de ce type indiquent si, oui ou non, il faut qu'ils soient les initiateurs du processus d'établissement d'AR.

CLIENT Les points d'extrémité de ce type émettent des APDU de demande de services confirmés vers les serveurs et reçoivent des APDU de réponse de services confirmés et une PDU de demande de services non confirmés.

SERVER Les points d'extrémité de ce type reçoivent des APDU de demande de services confirmés et non confirmés provenant des clients et émettent vers ceux-ci des APDU de réponse de services confirmés et une PDU de demande de services confirmés. Ils peuvent également émettre des APDU de demande de services non confirmés vers des clients.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Remote Address Configuration Type**

Cet attribut spécifie comment l'adresse distante de l'AREP est configurée. Les valeurs valides sont:

| | |
|---|---|
| FREE | La valeur “FREE” indique que la destination des PDU de la FAL issues de l'ARP source est fournie dynamiquement. |
| LINKED | La valeur “LINKED” indique que la destination des PDU de la FAL est configurée avec l'attribut RemoteDlsapAddress contenu dans le Mapping de DL (spécifié dans la CEI 61158-6-5). |

**Max Outstanding Requests Calling**

Cet attribut conditionnel indique le nombre maximal de réponses que l'AREP peut attendre de l'AREP distant.

**Max Outstanding Requests Called**

Cet attribut conditionnel indique le nombre maximal de réponses AREP que le point d'extrémité peut attendre de son utilisateur local.

**Initiator** (initiateur)

Cet attribut indique, lorsqu'il est TRUE, que le point d'extrémité a été configurée pour lancer l'établissement de l'AR. Il convient qu'un et un seul des AREP impliqués dans une AR soit l'initiateur. Lorsque cet AREP est un client, la valeur de cet attribut est toujours TRUE. Lorsque cet AREP est un serveur, la valeur de cet attribut est toujours FALSE. Lorsque cet AREP est un Peer (c'est-à-dire un homologue), la valeur de cet attribut peut être l'un ou l'autre, tant que les deux AREP dans l'AR ne sont pas configurés pour être l'initiateur.

**Remote AP Name**

Cet attribut facultatif spécifie le nom de l'AP attaché à l'AREP distant.

**Transmit DL Mapping Reference**

Cet attribut fournit une référence au mapping à la couche liaison de données sous-jacente pour le trajet d'acheminement en émission pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**Receive DL Mapping Reference**

Cet attribut fournit une référence au mapping de la couche liaison de données sous-jacente pour le trajet d'acheminement en réception pour cet AREP. Les mappings de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

#### 6.3.5.4 Services

**Confirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service confirmé.

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé.

**Establish**

Ce service est utilisé pour établir une AR.

**Abort**

Ce service est utilisé pour abandonner (Abort) une AR.

**Reject**

Ce service déclenché par un fournisseur est utilisé pour signaler la détection d'une erreur de protocole.

### 6.3.6 Spécification de la classe Buffered user-triggered bi-directional (BUB) AR endpoint

#### 6.3.6.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge l'échange, à la demande, de services confirmés et non confirmés entre deux processus application.

Le comportement de ce type d'AR peut être décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande ou de réponse la présente à son AREP comme une unité de données de service d'ASE AR. L'AREP qui envoie l'APDU de demande ou de réponse l'écrit dans le buffer de la couche liaison de données, remplaçant complètement le contenu existant du buffer. La couche liaison de données transfère le contenu du buffer à la prochaine occasion de transfert. Le buffer peut également être transféré chaque fois qu'une demande d'émettre (AR Compel) est reçue, localement ou en provenance du réseau.

L'AREP qui reçoit l'APDU de demande ou de réponse provenant de sa couche sous-jacente la met en buffer en vue de sa livraison. Si l'APDU est une réponse, le point d'extrémité arrête son temporisateur associé.

Si l'AREP qui envoie l'APDU reçoit une autre APDU avant que le contenu du buffer ne soit émis, le contenu du buffer sera remplacé par la nouvelle APDU et la précédente APDU sera perdue. Lorsque le contenu du buffer est émis, l'ASE AR notifie l'émission à l'utilisateur.

Au point d'extrémité destinataire, l'APDU est reçue en provenance du réseau et est écrite immédiatement dans le buffer, remplaçant complètement le contenu existant du buffer. Le point d'extrémité notifie à l'utilisateur que l'APDU est arrivée et la livre à l'utilisateur en fonction de l'interface utilisateur locale. Si l'APDU n'a pas été livrée avant que n'arrive la prochaine APDU, elle sera remplacée par la prochaine APDU et sera perdue.

Un utilisateur de la FAL qui reçoit l'émission tamponnée peut ultérieurement demander de recevoir l'APDU actuellement placée en tampon.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | Client<br>Server<br>Homologue |
| Cardinalité: | 1 à 1 |
| Cohérence temporelle: | Non |

#### 6.3.6.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | | | Buffered User-triggered Bi-directional AREP |
| **CLASS ID:** | | | 42 |
| **PARENT CLASS:** | | | AR Endpoint |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1 | (m) | Attribut: | Role (CLIENT, SERVER, PEER) |
| 2 | (m) | Attribut: | AREP State |
| 3 | (m) | Attribut: | Confirmed Services Flag(TRUE/FALSE) |
| 4 | (m) | Attribut: | Initiator (TRUE, FALSE) |
| 5 | (m) | Attribut: | Transmit DL Mapping Reference |
| 6 | (m) | Attribut: | Receive DL Mapping Reference |
| 7 | (c) | Contrainte: | Confirmed Services = TRUE |
| 7.1 | (m) | Attribut: | Max Outstanding Requests Calling |
| 7.2 | (m) | Attribut: | Max Outstanding Requests Called |
| 8 | (m) | Attribut: | Remote AP Name |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Unconfirmed Send |
| 2 | (o) | OpsService: | AR Compel |
| 3 | (o) | OpsService: | Confirmed Send |
| 4 | (o) | OpsService: | Establish |
| 5 | (o) | OpsService: | Get Buffered Message |

#### 6.3.6.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PEER (CLIENT/SERVER) Les points d'extrémité de ce type peuvent agir comme clients, comme serveurs ou simultanément comme les deux. Il convient que les points d'extrémité de ce type indiquent si, oui ou non, il faut qu'ils soient les initiateurs du processus d'établissement d'AR.

CLIENT Les points d'extrémité de ce type émettent des APDU de demande de services confirmés et non confirmés vers les serveurs et reçoivent des APDU de réponse de services confirmés.

SERVER Les points d'extrémité de ce type reçoivent des APDU de demande de services confirmés et non confirmés provenant des clients et émettent vers ceux-ci des APDU de réponse de services confirmés. Ils peuvent également émettre des APDU de demande de services non confirmés vers des clients.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Confirmed Services Flag**

Cet attribut spécifie, lorsqu'il est TRUE, que ce point d'extrémité prend en charge l'acheminement de demandes de services confirmés.

**Initiator**

Cet attribut indique, lorsqu'il est TRUE, que le point d'extrémité a été configurée pour lancer l'établissement de l'AR. Il convient qu'un et un seul des AREP impliqués dans une AR soit l'initiateur. Lorsque cet AREP est un client, la valeur de cet attribut est toujours TRUE. Lorsque cet AREP est un serveur, la valeur de cet attribut est toujours FALSE. Lorsque cet AREP est un Peer (c'est-à-dire un homologue), la valeur de cet attribut peut être l'un ou l'autre, tant que les deux AREP de l'AR ne sont pas configurés pour être l'initiateur.

**Transmit DL Mapping Reference**

Cet attribut fournit une référence au mapping à la couche liaison de données sous-jacente pour le trajet d'acheminement en émission pour cet AREP. Les références du mapping de DL pour les AREP sont spécifiées dans la CEI 61158-6-5.

**Receive DL Mapping Reference**

Cet attribut qui fournit le mapping à la couche sous-jacente pour le trajet d'acheminement en réception pour cet AREP. Les références du mapping de DL pour les AREP sont spécifiées dans la CEI 61158-6-5.

**Max Outstanding Requests Calling**

Cet attribut conditionnel indique le nombre maximal de réponses que l'AREP peut attendre de l'AREP distant. Cet attribut n'est présent que si ce point d'extrémité prend en charge les services confirmés (confirmed services = TRUE).

**Max Outstanding Requests Called**

Cet attribut conditionnel indique le nombre maximal de réponses AREP que le point d'extrémité peut attendre de son utilisateur local. Cet attribut n'est présent que si ce point d'extrémité prend en charge les services confirmés (Confirmed Services = TRUE).

**Remote AP Name**

Cet attribut spécifie l'AP attaché à l'AREP distant.

#### 6.3.6.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé.

**AR Compel**

Ce service facultatif est utilisé pour amener une APDU mise en tampon, localement ou à distance, en vue de son émission dans la couche liaison de données à être transférée à la prochaine occasion disponible.

**Confirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service confirmé.

**Establish**

Ce service est utilisé pour établir une AR.

**Get Buffered Message**

Ce service local est utilisé pour recevoir une APDU provenant du buffer utilisé par une AR.

### 6.3.7 Spécification de la classe Buffered network-scheduled uni-directional (BNU) AR endpoint

#### 6.3.7.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge le modèle "push" (en poussée) pour la distribution bufférisée programmée de services non confirmés vers un ou plusieurs processus application.

Le comportement de ce type d'AR peut être décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande ou de réponse la présente comme une unité de données de service d'ASE AR à son AREP pour sa distribution. L'AREP qui envoie l'APDU de demande ou de réponse l'écrit dans le buffer de la couche liaison de données, remplaçant complètement le contenu existant du buffer. La couche liaison de données transfère le contenu du buffer à la prochaine occasion de transfert programmé. Le buffer est également transféré chaque fois qu'une demande d'émettre (AR Compel) est reçue, localement ou en provenance du réseau.

NOTE Les demandes non programmées d'émettre n'ont aucun effet sur le comportement de ce type d'AR. Elles sont traitées comme étant "hors bande" par rapport au comportement de l'AR.

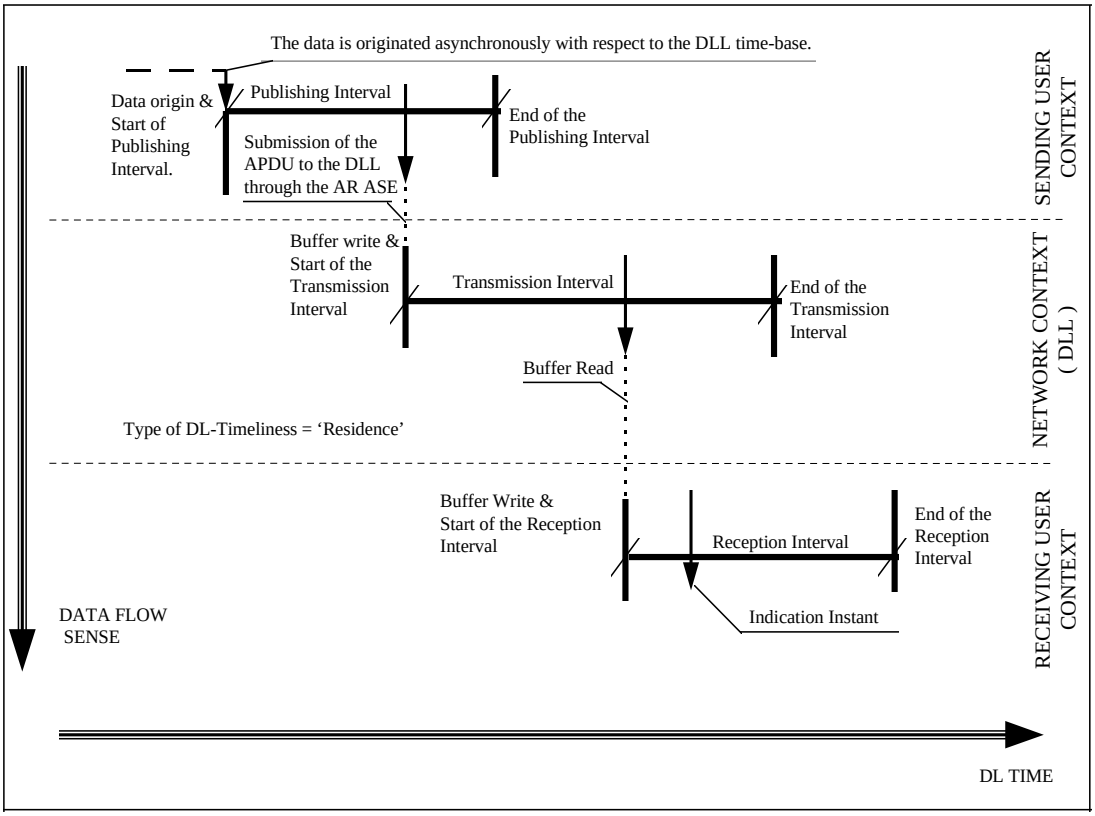
Si l'AREP qui envoie l'APDU reçoit une autre APDU avant que le contenu du buffer ne soit émis, le contenu du buffer sera remplacé par la nouvelle APDU et la précédente APDU sera perdue. Lorsque le contenu du buffer est émis, l'ASE AR notifie l'émission à l'utilisateur.

Au point d'extrémité destinataire, l'APDU est reçue en provenance du réseau et est écrite immédiatement dans le buffer, écrasant complètement en écriture le contenu existant du buffer. Le point d'extrémité notifie à l'utilisateur que l'APDU est arrivée et la livre à l'utilisateur en fonction de l'interface utilisateur locale. Si l'APDU n'a pas été livrée avant que n'arrive la prochaine APDU, elle sera écrasée en écriture par la prochaine APDU et sera perdue.

Un utilisateur de la FAL qui reçoit l'émission bufférisée peut ultérieurement demander de recevoir l'APDU actuellement placée dans le buffer.

Ce type d'AR peut également prendre en charge, en option, la capacité de l'ASE AR de déterminer la cohérence temporelle d'édition, d'émission ou de réception. Pour la cohérence temporelle d'édition, l'ASE AR expéditeur marquera le buffer s'il n'a pas été mis à jour dans l'intervalle spécifié de mise à jour. Pour la cohérence temporelle de réception, l'ASE AR destinataire marquera le buffer s'il n'a pas été reçu dans l'intervalle spécifié de réception. Les informations relatives à la cohérence temporelle tant d'édition que de réception sont données. Pour la cohérence temporelle d'émission, l'ASE AR expéditeur détermine si le buffer a été émis dans l'intervalle spécifié. Si l'émission n'a pas lieu, l'utilisateur de la FAL en reçoit notification et le contenu du buffer est mis à jour pour indiquer le manque de cohérence temporelle d'émission.

La cohérence temporelle "residence" est prise en charge par ce type d'AREP. La Figure 5 illustre ce type de cohérence temporelle.




**Légende**

| Anglais | Français |
|---|---|
| SENDING USER CONTEXT | CONTEXTE DE L'UTILISATEUR EXPÉDITEUR |
| NETWORK CONTEXT (DLL) | CONTEXTE DU RESEAU (DLL) |
| The data is originated asynchronously with respect to the DLL time-base | Les données sont émises de façon asynchrone par rapport à la base de temps de la DLL. |
| Publishing Interval | Intervalle de production |
| End of the Publishing Interval | Fin de l'intervalle de production |
| Data origin & Start of Publishing Interval | Origine des données et Début de l'Intervalle de production |
| Submission of the APDU to the DLL through the AR AS | Présentation de l'APDU à la DLL par l'intermédiaire de l'ASE AR |
| Buffer Write &Start of the Reception Interval | Écriture de buffer et Début de l'intervalle de réception |
| Transmission Interval | Intervalle d'émission |
| End of the Transmission Interval | Fin de l'intervalle d'émission |
| Buffer Read | Lecture de buffer |
| Type of DL-Timeliness = 'Residence' | Type de cohérence temporelle de DL = 'Residence' |
| Buffer Write &Start of the Transmission Interval | Écriture de buffer et Début de l'intervalle d'émission |
| Reception Interval | Intervalle de réception |
| End of the reception Interval | Fin de l'intervalle de réception |

| Anglais | Français |
|---|---|
| Indication Instant | Instant d'indication |
| RECEIVING USER CONTEXT | CONTEXTE DE L'UTILISATEUR RECEPTEUR |
| DATA FLOW SENSE | SENS DU FLUX DES DONNÉES |
| DL TIME | TEMPS DL |

**Figure 5 – Cohérence temporelle "residence"**

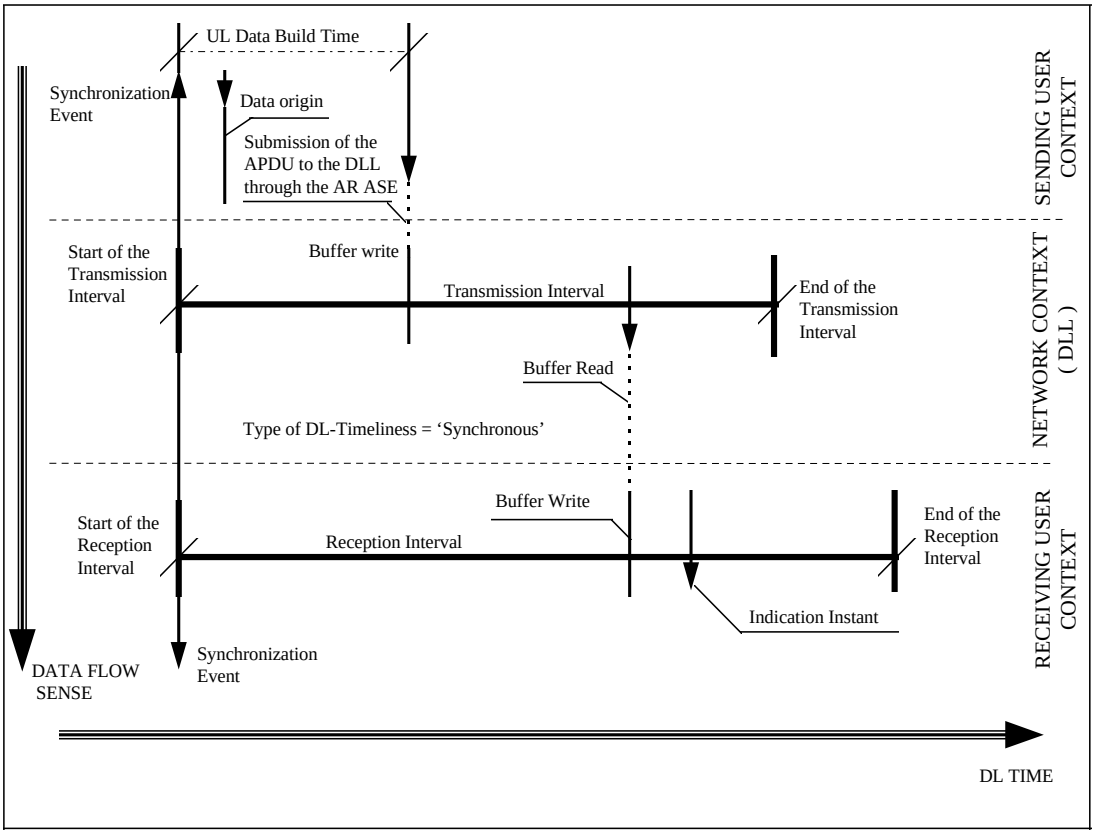
Ce type de cohérence temporelle requiert l'utilisation d'un temporisateur de liaison de données associé à un buffer de données (en émission ou en réception). Le temporisateur est déclenché lorsque les données sont écrites dans le buffer d'émission par l'AREP éditeur ou sont reçues en provenance du réseau dans le buffer de l'abonné. La longueur de l'intervalle de temps est configurée pour l'AREP par la gestion.

L'indicateur de cohérence temporelle pour ce type d'AREP est calculé comme suit:

Les données d'AREP éditeur sont opportunes si elles sont émises dans la fenêtre de cohérence temporelle après avoir été placées dans le buffer de liaison de données en vue de leur émission. Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les émissions ultérieures des mêmes données du buffer indiqueront son statut courant.

Les données d'AREP abonné sont opportunes si elles sont reçues comme étant opportunes par la couche liaison de données et sont lues dans le buffer de réception par l'AREP dans les limites de la fenêtre de cohérence temporelle qui avait été énoncée lorsque les données étaient reçues. Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les lectures ultérieures dans le buffer des mêmes données indiqueront son statut courant.

La cohérence temporelle "synchronized" est prise en charge par ce type d'AREP. La Figure 6 illustre ce type de cohérence temporelle.




**Légende**

| Anglais | Français |
|---|---|
| UL Data Build Time | Temps de construction des données UL |
| SENDING USER CONTEXT | CONTEXTE DE L'UTILISATEUR EXPEDITEUR |
| NETWORK CONTEXT (DLL) | CONTEXTE DU RESEAU (DLL) |
| RECEIVING USER CONTEXT | CONTEXTE DE L'UTILISATEUR RECEPTEUR |
| Synchronization Event | Événement de synchronisation |
| Data origin | Origine des données |
| Submission of the APDU to the DLL through the AR ASE | Présentation de l'APDU à la DLL par l'intermédiaire de l'ASE AR |
| Buffer write | Écriture du buffer |
| Start of the Transmission Interval | Début de l'intervalle d'émission |
| Transmission Interval | Intervalle d'émission |
| End of the Transmission Interval | Fin de l'intervalle d'émission |
| Buffer Read | Lecture du buffer |
| Type of DL-Timeliness = "Synchronous" | Type de cohérence temporelle DL = "Synchronous" (synchrone) |
| Start of the Reception Interval | Début de l'intervalle de réception |
| Reception Interval | Intervalle de réception |
| Buffer Write | Écriture de buffer |
| End of the Reception Interval | Fin de l'Intervalle de réception |
| Synchronization Event | Événement de synchronisation |
| DATA FLOW SENSE | SENS DU FLUX DE DONNÉES |
| DL TIME | TEMPS DL |
| Indication instant | Instant d'indication |

**Figure 6 – Cohérence temporelle "synchronized"**

Ce type de cohérence temporelle requiert l'utilisation de deux connexions de liaison de données en soutien. L'une est utilisée pour éditer les données et l'autre est utilisée pour distribuer une marque de synchronisation (“sync mark”) au sein de la couche liaison de données vers chacun des points d'extrémité participant comme éditeur ou abonné. La marque de synchronisation fait démarrer un temporisateur associé à un buffer de données (en émission ou en réception) dans la couche liaison de données. La longueur de l'intervalle de temps est configurée pour l'AREP par la gestion.

L'indicateur de cohérence temporelle pour ce type d'AREP est calculé comme suit:

Les données d'AREP éditeur sont opportunes si elles sont mises dans le buffer dans les limités de la fenêtre temporelle après la réception d'une “sync mark”. Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les émissions ultérieures des mêmes données du buffer indiqueront le statut courant du buffer.

Les données d'AREP abonné sont opportunes si elles sont reçues comme étant opportunes par la couche liaison de données et sont reçues dans les limites de la fenêtre temporelle. Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les données reçues après l'expiration de la fenêtre, mais avant la prochaine marque de synchronisation, sont également considérées comme étant NON opportunes. Les lectures ultérieures dans le buffer des mêmes données indiqueront son statut courant.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | Éditeur Push<br>Abonné Push |
| Cardinalité: | 1 à plusieurs |
| Cohérence temporelle: | Facultatif |

#### 6.3.7.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | | | Buffered Network-Scheduled Uni-directional AR Endpoint |
| **CLASS ID:** | | | 38 |
| **PARENT CLASS:** | | | AR Endpoint |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1 | (m) | Attribut: | Role (PUSH PUBLISHER, PUSH SUBSCRIBER) |
| 2 | (m) | Attribut: | AREP State |
| 2 | (c) | Contrainte | Role == PUSH SUBSCRIBER |
| 2.1 | (o) | Attribut: | Duplicate Pdu Detection Supported |
| 3 | (m) | Attribut: | DL Mapping Reference |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Unconfirmed Send |
| 2 | (o) | OpsService: | AR Compel |
| 3 | (o) | OpsService: | Get Buffered Message |
| 4 | (o) | OpsService: | Schedule Communication |
| 5 | (o) | OpsService: | Cancel Scheduled Sequence |

#### 6.3.7.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PUSH-PUBLISHER Les points d'extrémité de ce type éditent leurs données en émettant des APDU de demande de services non confirmés.

PUSH-SUBSCRIBER Les points d'extrémité de ce type reçoivent des données produites dans des APDU de réponse de services non confirmés

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**DL Mapping Reference**

Pour des AREP PUSH-PUBLISHER, cet attribut spécifie le mapping du trajet d'acheminement de l'émission. Pour des AREP PUSH-SUBSCRIBER, cet attribut spécifie le mapping du trajet d'acheminement de laréception. Les attributs de mapping de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

#### 6.3.7.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé.

**AR Compel**

Ce service facultatif est utilisé pour qu'une APDU bufferisée localement pour une émission (pour des AREP PUSH-PUBLISHER) ou bufferisée à distance pour une émission (pour des AREP PUSH-SUBSCRIBER) dans la couche liaison de données soit émise à la prochaine occasion disponible.

**Get Buffered Message**

Ce service local est utilisé pour recevoir une APDU provenant du buffer utilisé par une AR.

**Schedule Communication**

Ce service local donne à l'utilisateur de l'AL la capacité de programmer une séquence de DL-COMPEL pour l'AREP.

**Cancel Scheduled Sequence**

Ce service local donne à l'utilisateur de l'AL la capacité d'annuler une séquence existante qui avait été programmée précédemment pour l'AREP.

### 6.3.8 Spécification de la classe Buffered network-scheduled bi-directional (BNB) AR endpoint

#### 6.3.8.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge l'échange cyclique de services confirmés (Read seulement) et l'échange de la demande de services non confirmés entre deux processus application utilisant des buffers. Le cycle est donné par le réseau.

Le comportement de ce type d'AR peut être décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande de service Read (lecture) ou une APDU de demande de services non confirmés la présente comme une unité de données de service d'ASE AR à son AREP. L'AREP qui envoie l'APDU de demande l'écrit dans le buffer de couche liaison de données, remplaçant complètement le contenu existant du buffer. De surcroît, une copie de l'APDU est stockée dans un buffer local au sein de l'AREP si le buffer de couche liaison de données est actuellement utilisé par une demande précédente. La couche liaison de données transfère le contenu du buffer en une fois à la première occasion de transfert lorsque la demande à émettre (AR Compel) est reçue du réseau.

Au point d'extrémité destinataire, l'APDU est reçue en provenance du réseau et elle est écrite dans le buffer, remplaçant complètement le contenu existant du buffer. Le point d'extrémité notifie à l'utilisateur que l'APDU confirmée (Read) est arrivée et la livre à l'utilisateur en fonction de l'interface utilisateur locale. De surcroît, une copie de l'APDU est stockée dans un buffer local au sein de l'AREP. La réponse suivante provenant de l'utilisateur d'ASE AR est écrite par l'AREP dans le buffer de la couche liaison de données, remplaçant complètement le contenu existant du buffer. La couche liaison de données transfère le contenu du buffer en une fois à la première occasion de transfert lorsque la demande à émettre (AR Compel) est reçue du réseau. L'utilisateur d'ASE AR reçoit de nouveau une notification avec la demande restaurée lorsque la réponse est envoyée sur le réseau. Cela amène l'utilisateur d'ASE AR à répondre à nouveau, de façon cyclique, avec la variable courante demandée.

Au point d'extrémité demandeur, l'APDU contenant la réponse reçue en provenance du réseau, elle est écrite dans le buffer, remplaçant complètement le contenu existant du buffer. Le point d'extrémité notifie à l'utilisateur que l'APDU est arrivée et la livre à l'utilisateur en fonction de l'interface utilisateur local comme étant la confirmation. Si l'utilisateur d'ASE AR souhaite acheminer de nouveau une demande Read, la réponse est générée localement à partir du buffer de la couche liaison de données. Le contenu du buffer est mis à jour, de façon cyclique, par le réseau avec le mécanisme connexe en l'extrémité destinataire.

L'AREP qui reçoit l'APDU de demande non confirmée provenant de sa couche sous-jacente la met en file d'attente en vue de sa distribution à son utilisateur d'ASE AR dans l'ordre dans lequel elle a été reçue.

De plus, l'AREP surveille la connexion en observant les activités du réseau.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | Homologue |
| Cardinalité: | 1 à 1 |
| Cohérence temporelle: | Non |

#### 6.3.8.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | Buffered Network-Triggered Bi-directional AR Endpoint | | |
| **CLASS ID:** | | | 44 |
| **PARENT CLASS:** | | | AR Endpoint |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1 | (m) | Attribut: | Role (CLIENT, SERVER, PEER) |
| 2 | (m) | Attribut: | AREP State |
| 3 | (m) | Attribut: | Remote Address Configuration Type (FREE,LINKED) |
| 4 | (m) | Attribut: | Initiator (TRUE/FALSE) |
| 5 | (m) | Attribut: | Receive DL Mapping Reference |
| 6 | (m) | Attribut: | Transmit DL Mapping Reference |
| 7 | (o) | Attribut: | Remote AP Name |
| 8 | (m) | Attribut: | CIN |
| **SERVICES:** | | | |
| 1 | (o) | OpsService: | Unconfirmed Send |
| 2 | (o) | OpsService: | Confirmed Send |
| 3 | (o) | OpsService: | Establish |
| 4 | (o) | OpsService: | Abort |

#### 6.3.8.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

CLIENT, SERVER, PEER () Les points d'extrémité de ce type peuvent agir comme clients, comme serveurs ou comme les deux simultanément. Il convient que les points d'extrémité de ce type indiquent si, oui ou non, il faut qu'elles soient les initiatrices du processus d'établissement d'AR.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Remote Address Configuration Type**

Cet attribut spécifie comment l'adresse distante de l'AREP est configurée. Les valeurs valides sont:

FREE La valeur "FREE" indique que la destination des PDU de la FAL issues de l'ARP source est fournie dynamiquement dans la demande de service.

LINKED La valeur "LINKED" indique que la destination des PDU de la FAL est configurée avec l'attribut RemoteDlsapAddress contenu dans le Mapping de DL (spécifié dans la CEI 61158-6-5).

**Initiator** (initiateur)

Cet attribut indique, lorsqu'il est TRUE, que le point d'extrémité a été configurée pour lancer l'établissement de l'AR. Il convient qu'un et un seul des AREP impliqués dans une AR soit l'initiateur. Lorsque cet AREP est un client, la valeur de cet attribut est toujours TRUE. Lorsque cet AREP est un serveur, la valeur de cet attribut est toujours FALSE. Lorsque cet AREP est un Peer (c'est-à-dire un homologue), la valeur de cet attribut peut être l'un ou l'autre, tant que les deux AREP de l'AR ne sont pas configurés pour être l'initiateur.

**Receive DL Mapping Reference**

Cet attribut fournit une référence au mapping de la couche liaison de données sous-jacente pour le trajet d'acheminement à partir du gestionnaire d'édition (Publishing Manager). Les attributs de mapping de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**Transmit DL Mapping Reference**

Cet attribut conditionnel fournit une référence au mapping à la couche liaison de données sous-jacente pour le trajet d'acheminement en émission. Il n'est présent que pour des PULL PUBLISHER (éditeurs pull) et des PULL PUBLISHING MANAGER (gestionnaires de production pull). Les attributs de mapping de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

**Remote AP Name**

Cet attribut facultatif spécifie le nom de l'AP attaché à l'AREP distant.

**CIN**

Cet attribut (CIN – Control Interval Network-triggered, c'est-à-dire Intervalle de commande pour connexion déclenchée par un réseau) indique l'intervalle de commande pour surveiller une connexion déclenchée par un réseau.

#### 6.3.8.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé.

**Confirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service confirmé.

**Establish**

Ce service est utilisé pour établir une AR.

**DeEstablish**

Ce service facultatif est utilisé pour désinstaller (mettre progressivement fin à) une AR.

**Abort**

Ce service est utilisé pour mettre fin à une AR.

### 6.3.9 Specification de la classe Buffered network-scheduled and unscheduled uni-directional (BNU-Mp) AR endpoint

#### 6.3.9.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge le modèle "push" (en poussée) pour la distribution bufférisée programmée de services non confirmés vers un ou plusieurs processus application.

Le comportement de ce type d'AR peut être décrit comme suit.

Le modèle push est utilisé pour le trafic programmé, comme suit.

Un utilisateur d'ASE AR souhaitant diffuser de nouvelles données utilise une unité de données de service d'ASE AR avec un codage spécial (voir CEI 61158-6-5) en son AREP pour distribution. L'AREP qui envoie les données les écrit dans le buffer de la couche liaison de données, remplaçant complètement le contenu existant du buffer. La couche liaison de données transfère le contenu du buffer à la prochaine occasion programmée. Lorsque le contenu du buffer est émis, l'ASE AR notifie l'émission à l'utilisateur.

Si l'AREP qui envoie les données reçoit, de l'utilisateur d'ASE AR, d'autres données avant que le contenu du buffer ne soit émis, le contenu du buffer sera remplacé par les nouvelles données et les données précédentes seront perdues.

Au point d'extrémité destinataire, les données sont reçues en provenance du réseau et elles sont écrites immédiatement dans le buffer, écrasant complètement en écriture le contenu existant du buffer. Le point d'extrémité notifie à l'utilisateur que les données sont arrivées et les livre à l'utilisateur. Des lectures locales supplémentaires du contenu du buffer sont possibles. Si les données n'ont pas été livrées avant que n'arrivent les prochaines données, elles seront écrasées en écriture par les prochaines données et seront perdues.

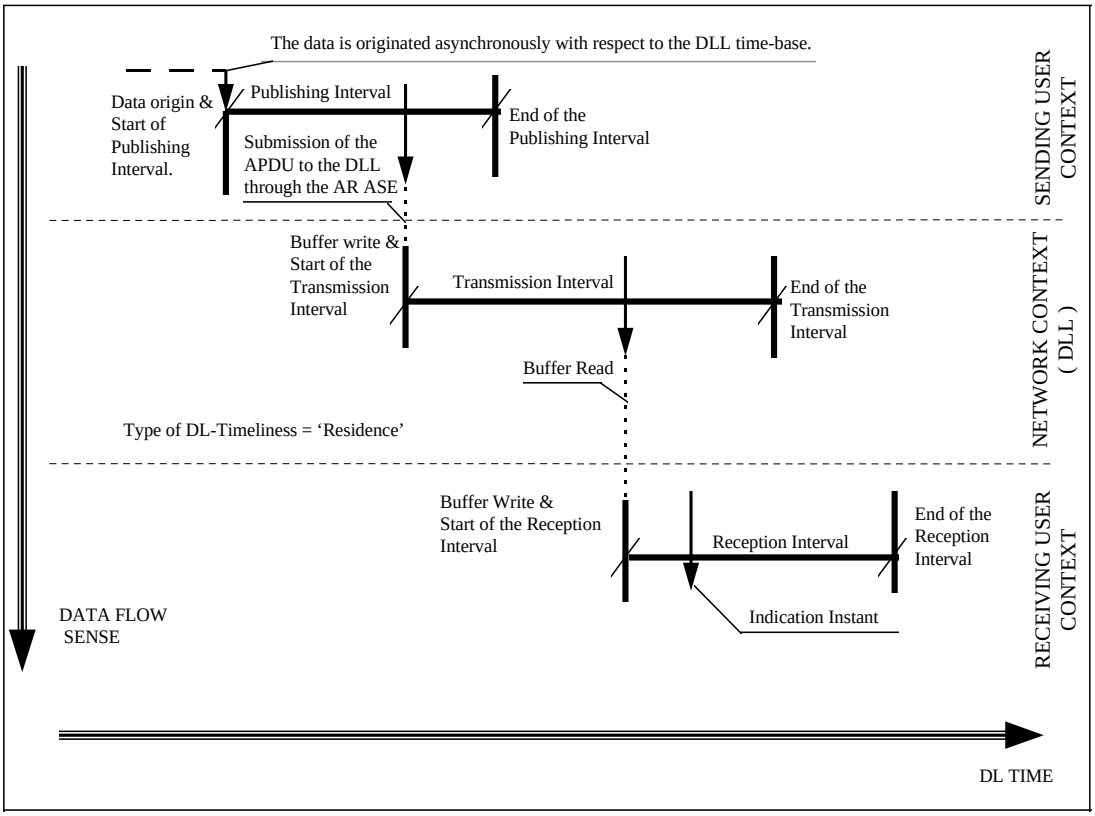
Le modèle pull est utilisé pour un trafic à la demande, comme suit.

Un utilisateur d'ASE AR souhaitant envoyer ou lire la valeur distante du buffer utilise les services Remote-Write ou Remote-Read avec une unité de données de données d'ASE AR spécialement codée (voir CEI 61158-6-5). Un utilisateur d'ASE AR souhaitant diffuser des données à la demande les écrit dans le buffer de la couche liaison de données en remplaçant le contenu existant du buffer. La couche liaison de données transfère le contenu du buffer après avoir sollicité une occasion d'émission. L'émission du buffer est rapportée à l'utilisateur comme étant la confirmation du service. Pour le point d'extrémité destinataire, les données sont reçues en provenance du réseau et sont écrites immédiatement dans le buffer, écrasant complètement le contenu existant du buffer. Le point d'extrémité notifie à l'utilisateur que les données sont arrivées et les livre à l'utilisateur. Des lectures locales supplémentaires du contenu du buffer sont possibles.

Un utilisateur d'ASE AR souhaitant lire la valeur distante d'un buffer demande à la couche liaison de données de solliciter une occasion d'émission pour obtenir la valeur du buffer distant. L'émission du buffer est rapportée à l'utilisateur comme étant la confirmation du service. Au point d'extrémité destinataire, les données sont reçues en provenance du réseau et sont écrites immédiatement dans le buffer, écrasant complètement le contenu existant du buffer. Le point d'extrémité notifie à l'utilisateur que les données sont arrivées et les livre à l'utilisateur. Des lectures locales supplémentaires du contenu du buffer sont possibles.

Ce type d'AR peut également prendre en charge, en option, la capacité de l'ASE AR de déterminer la cohérence temporelle d'édition, d'émission ou de réception. Pour la cohérence temporelle d'édition, l'ASE AR expéditeur marquera le buffer s'il n'a pas été mis à jour dans l'intervalle spécifié de mise à jour. Pour la cohérence temporelle de réception, l'ASE AR destinataire marquera le buffer s'il n'a pas été reçu dans l'intervalle spécifié de réception. Les informations relatives à la cohérence temporelle tant d'édition que de réception sont données. Pour la cohérence temporelle d'émission, l'ASE AR expéditeur détermine si le buffer a été émis dans l'intervalle spécifié. Si l'émission n'a pas lieu, l'utilisateur de la FAL en reçoit notification et le contenu du buffer est mis à jour pour indiquer le manque de cohérence temporelle d'émission.

La cohérence temporelle "residence" est prise en charge par ce type d'AREP. La Figure 7 illustre ce type de cohérence temporelle.




**Légende**

| Anglais | Français |
|---|---|
| The data is originated asynchronously with respect to the DLL time-base | Les données sont émises de façon asynchrone par rapport à la base de temps de la DLL |
| SENDING USER CONTEXT | CONTEXTE DE L'UTILISATEUR EXPEDITEUR |
| NETWORK CONTEXT (DLL) | CONTEXTE DU RESEAU (DLL) |
| RECEIVING USER CONTEXT | CONTEXTE DE L'UTILISATEUR RECEPTEUR |
| Data origin & Start of Publishing Interval | Origine des données et Début de l'Intervalle de Publication |
| Publishing Interval | Intervalle d'édition |
| Submission of the APDU to the DLL through the AR ASE | Présentation de l'APDU à la DLL par l'intermédiaire de 'ASE AR |
| End of Publishing Interval | Fin de l'intervalle d'édition |
| Buffer write & Start of the Transmission Interval | Écriture du buffer et Début de l'intervalle d'émission |

| Anglais | Français |
|---|---|
| Transmission Interval | Intervalle d'émission |
| End of the Transmission Interval | Fin de l'Intervalle d'émission |
| Buffer Read | Lecture du buffer |
| Type of DL-Timeliness = 'Residence' | Type de cohérence temporelle de DL = "Residence" |
| Buffer write & Start of the Reception Interval | Écriture du buffer et Début de l'Intervalle de réception |
| Reception Interval | Intervalle de réception |
| Indication Instant | Instant d'indication |
| End of the Reception Interval | Fin de L'intervalle de réception |
| DATA FLOW SENSE | SENS DU FLUX DES DONNÉES |
| DL TIME | TEMPS de DL |

**Figure 7 – Cohérence temporelle "residence"**

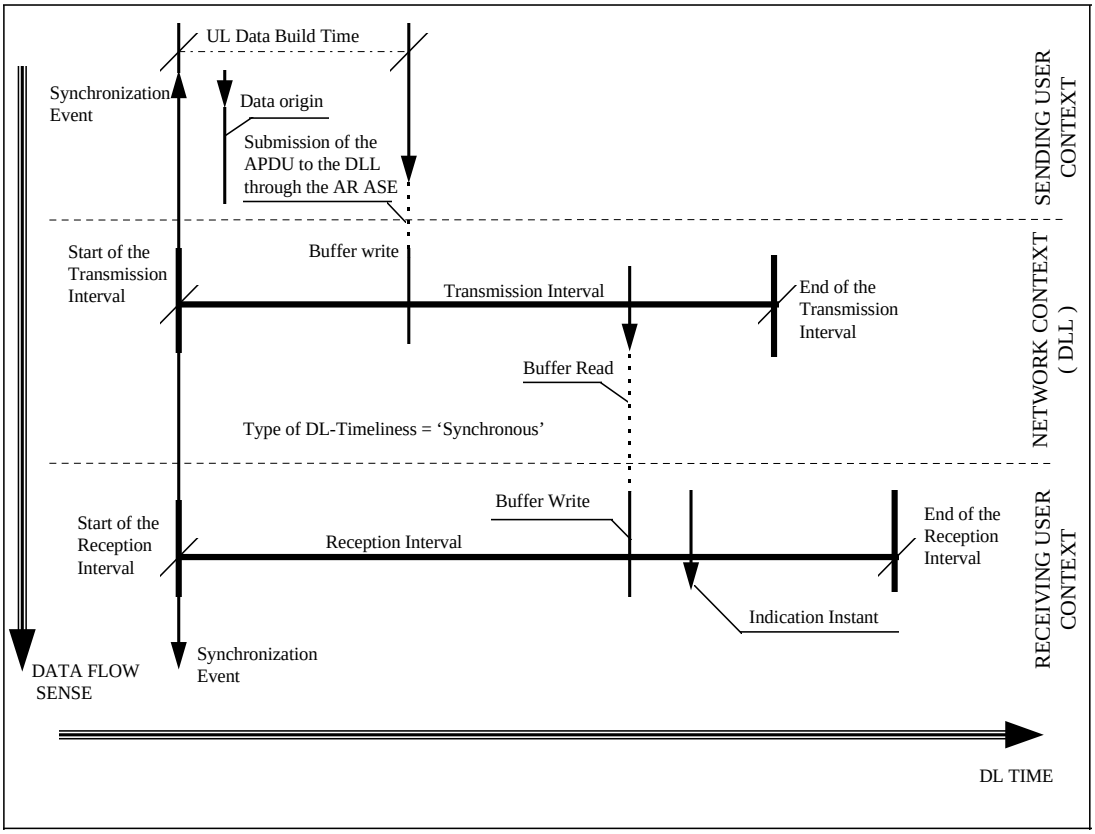
Ce type de cohérence temporelle requiert l'utilisation d'un temporisateur de liaison de données associé à un buffer de données (en émission ou en réception). Le temporisateur est déclenché lorsque les données sont écrites dans le buffer d'émission par l'AREP éditeur ou sont reçues en provenance du réseau dans le buffer de l'abonné. La longueur de l'intervalle de temps est configurée pour l'AREP par la gestion.

L'indicateur de cohérence temporelle pour ce type d'AREP est calculé comme suit.

Les données d'AREP éditeur sont opportunes si elles sont émises dans la fenêtre de cohérence temporelle après avoir été placées dans le buffer de liaison de données en vue de leur émission. Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les émissions ultérieures des mêmes données du buffer indiqueront son statut courant.

Les données d'AREP abonné sont opportunes si elles sont reçues comme étant opportunes par la couche liaison de données et sont lues dans le buffer de réception par l'AREP dans les limites de la fenêtre de cohérence temporelle qui avait été énoncée lorsque les données étaient reçues. Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les lectures ultérieures dans le buffer des mêmes données indiqueront son statut courant.

La cohérence temporelle "synchronized" est prise en charge par ce type d'AREP. La Figure 8 illustre ce type de cohérence temporelle.




**Légende**

| Anglais | Français |
|---|---|
| UL Data Build Time | Temps de construction des données UL |
| SENDING USER CONTEXT | CONTEXTE DE L'UTILISATEUR EXPEDITEUR |
| NETWORK CONTEXT (DLL) | CONTEXTE DU RESEAU (DLL) |
| RECEIVING USER CONTEXT | CONTEXTE DE L'UTILISATEUR RECEPTEUR |
| Synchronization Event | Événement de synchronisation |
| Data origin | Origine des données |
| Submission of the APDU to the DLL through the AR ASE | Présentation de l'APDU à la DLL par l'intermédiaire de l'ASE AR |
| Start of the Transmission Interval | Début de l'intervalle d'émission |
| Transmission Interval | Intervalle d'émission |
| End of the Transmission Interval | Fin de l'Intervalle d'émission |
| Buffer Read | Lecture du buffer |
| Type of DL-Timeliness = "Synchronous" | Type de cohérence temporelle DL = "Synchronous" (synchrone) |
| Start of the Reception Interval | Début de l'Intervalle de réception |
| Reception Interval | Intervalle de réception |
| Buffer write | Écriture du buffer |
| Indication Instant | Instant d'indication |

| Anglais | Français |
|---|---|
| End of the reception Interval | Fin de l'intervalle de réception |
| Indication Instant | Instant d'indication |
| Synchronization Event | Événement de synchronisation |
| DATA FLOW SENSE | Sens du flux de données |
| DL TIME | TEMPS de DL |

**Figure 8 – Cohérence temporelle "synchronized"**

Ce type de cohérence temporelle requiert l'utilisation de deux connexions de liaison de données en soutien. L'une est utilisée pour produire les données et l'autre est utilisée pour distribuer une marque de synchronisation ("sync mark") au sein de la couche liaison de données vers chacun des points d'extrémité participant comme éditeur ou abonné. La marque de synchronisation fait démarrer un temporisateur associé à un buffer de données (en émission ou en réception) dans la couche liaison de données. La longueur de l'intervalle de temps est configurée pour l'AREP par la gestion.

L'indicateur de cohérence temporelle pour ce type d'AREP est calculé comme suit.

Les données d'AREP éditeur sont opportunes si elles sont mises dans le buffer dans les limites de la fenêtre temporelle après la réception d'une "sync mark". Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les émissions ultérieures des mêmes données du buffer indiqueront le statut courant du buffer.

Les données d'AREP éditeur sont opportunes si elles sont reçues comme étant opportunes par la couche liaison de données et sont reçues dans les limites de la fenêtre temporelle. Lorsque la fenêtre de cohérence temporelle expire, les données dans le buffer sont considérées comme étant NON opportunes. Les données reçues après l'expiration de la fenêtre, mais avant la prochaine marque de synchronisation, sont également considérées comme étant NON opportunes. Les lectures ultérieures dans le buffer des mêmes données indiqueront son statut courant.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

Rôles: Éditeur Push
Abonné Push

Cardinalité: 1 à plusieurs

Cohérence temporelle: Facultatif

#### 6.3.9.2 Modèle formel

**FAL ASE:** **ASE AR**
**CLASS:** Buffered Network-Scheduled Uni-directional AR Endpoint
**CLASS ID:** 39
**PARENT CLASS:** AR Endpoint
**ATTRIBUTS DE GESTION DE RÉSEAU**

| | | | |
|---|---|---|---|
| 1 | (m) | Attribut: | Role (PUSH PUBLISHER, PUSH SUBSCRIBER, PULL PUBLISHER, PULL SUBSCRIBER) |
| 2 | (m) | Attribut: | AREP State |
| 3 | (m) | Attribut: | DL Mapping Reference |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (o) | OpsService: | Unconfirmed Send |

| | | | |
|---|---|---|---|
| 2 | (o) | OpsService: | AR Compel |
| 3 | (o) | OpsService: | Get Buffered Message |
| 4 | (o) | OpsService: | Schedule Communication |
| 5 | (o) | OpsService: | Cancel Scheduled Sequence |
| 6 | (o) | OpsService: | Remote Read |
| 7 | (o) | OpsService: | Remote Write |

#### 6.3.9.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PUSH-PUBLISHER Les points d'extrémité de ce type produisent leurs données en émettant des APDU de demande de services non confirmés.

PUSH-SUBSCRIBER Les points d'extrémité de ce type reçoivent des données produites dans des APDU de réponse de services non confirmés

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**DL Mapping Reference**

Pour des AREP PUSH-PUBLISHER, cet attribut spécifie le mapping du trajet d'acheminement pour l'émission. Pour des AREP PUSH-SUBSCRIBER, cet attribut spécifie le mapping du trajet d'acheminement pour la réception. Les attributs de mapping de DL pour la couche liaison de données (CEI 61158-3-1) sont spécifiés dans la CEI 61158-6-5.

#### 6.3.9.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé. La structure de la PDU est spécifiée et décrite dans le Diagramme d'états du mapping de DL correspondant.

**AR Compel**

Ce service facultatif est utilisé de sorte qu'une APDU bufferisée localement pour émission (pour des AREP PUSH-PUBLISHER) bufferisée à distance pour émission (pour des AREP PUSH-SUBSCRIBER) dans la couche liaison de données à soit émise à la prochaine occasion disponible.

**Get Buffered Message**

Ce service local est utilisé pour recevoir une APDU provenant du buffer utilisé par une AR.

**Schedule Communication**

Ce service local donne à l'utilisateur de l'AL la capacité de programmer une séquence de DL-COMPEL pour l'AREP.

**Cancel Scheduled Sequence**

Ce service local donne à l'utilisateur de l'AL la capacité d'annuler une séquence existante qui avait été programmée précédemment pour l'AREP.

**Remote Read**

Ce service (basé sur le modèle Pull Subscriber) permet à un utilisateur de lire le contenu d'un buffer distant. La structure de la PDU est décrite et spécifiée dans le document de protocole par le Diagramme d'états du mapping de DL correspondant.

**Remote Write**

Ce service (basé sur le modèle Pull Subscriber) permet à un utilisateur d'écrire le contenu d'un buffer distant. La structure de la PDU est spécifiée et décrite dans le document de protocole par le Diagramme d'états du mapping de DL correspondant.

## 6.4 Résumé des classes de FAL

Le présent paragraphe contient un résumé des classes de FAL définies. Les valeurs de Class ID (identificateur de classe) ont été attribuées de manière à être compatibles avec les normes existantes.

Le Tableau 80 présente un résumé des classes.

**Tableau 80 – Résumé des classes de FAL**

| FAL ASE | Class | Class ID |
|---|---|---|
| Application Process | Application Process | 16 |
| | | |
| Data type | Fixed Length & String Data type | 5 |
| | Structure Data type | 6 |
| | Array Data type | 12 |
| | | |
| Application Relationship | AREP | 32 |
| | QUB-Co | 34 |
| | QUU | 36 |
| | BNU | 38 |
| | BNU Mp | 39 |
| | BUU (for future study) | 40 |
| | BUB | 42 |
| | BNB | 44 |
| | QUB-Cl | 45 |
| | QUB-FC | 46 |
| | QUB-Seg | 48 |
| | | |
| Variable | Fixed Length & String Variable | 7 |
| | Array Variable | 8 |
| | Data Structure Variable | 9 |
| | Variable List | 10 |
| | | |
| Event | Event | 4 |
| | Event List | 17 |
| | Notifier | 18 |
| | | |
| Load Region | Load Region | 2 |
| | | |
| Function Invocation | Function Invocation | 3 |
| | Action | 19 |

### 6.5 Services de FAL autorisés par le rôle de l'AREP

Le Tableau 81 ci-dessous définit les combinaisons valides de services et de rôles d'AREP (quels services APDU et quels AREP avec le rôle spécifié peuvent envoyer ou recevoir ?). Les colonnes "Unc" et "Cnf" indiquent si le service figurant dans la colonne de gauche est non confirmé (unconfirmed, Unc) ou confirmé (confirmed, Cnf).

**Tableau 81 – Services par rôle d'AREP**

| | Unc | Cnf | Client | | Serveur | | Éditeur push | | Abonné push | | Gesti. Édition Pull | | Editeur pull | | Abonné pull | | Report Src | | Report Sink | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Services de FAL** | | | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** | **req** | **rcv** |
| **ASE Mgt** | | | | | | | | | | | | | | | | | | | | |
| Create | | X | X | | X | | | | | | | | | | | | | | | |
| Delete | | X | X | | X | | | | | | | | | | | | | | | |
| Get Atributes | | X | X | | X | | | | | | | | | | | | | | | |
| Get Attribute List | | X | X | | X | | | | | | | | | | | | | | | |
| Set Attributes | | X | X | | X | | | | | | | | | | | | | | | |
| Begin Set Attributes | | X | X | | X | | | | | | | | | | | | | | | |
| End Set Attributes | | X | X | | X | | | | | | | | | | | | | | | |
| **ASE d'AP** | | | | | | | | | | | | | | | | | | | | |
| Subscribe | | X | X | | X | | | | | | | | | | | | | | | |
| Identify | | X | X | | X | | | | | | | | | | | | | | | |
| Get Status | | X | X | | X | | | | | | | | | | | | | | | |
| Status Notification | X | | | | | | X | | X | | | | | | | | X | | X | |
| Initiate | | X | X | | X | | | | | | | | | | | | | | | |
| Terminate (Mettre fin) | | X | X | | X | | | | | | | | | | | | | | | |
| Conclude | | X | X | | X | | | | | | | | | | | | | | | |
| Reject | | X | | X | | | | | | | | | | | | | | | | |
| **ASE AR** | | | | | | | | | | | | | | | | | | | | |
| AR-Confirmed Send | | | X | | X | | | | | | | | | | | | | | | |
| AR-Unconfirmed Send | | | X | X | X | X | X | | X | | | | | | | | | | | |
| AR-Establish | | | X | | X | | | | | | | | | | | | | | | |
| AR-De-Establish | | | 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-Schedule Communication | | | | | | | X | | X | | | | | | | | | | | |
| AR-Cancel Schedule Sequence | | | | | | | X | | X | | | | | | | | | | | |
| AR-Status | | | X | | X | | | | | | | | | | | | | | | |
| AR-XON-OFF | | | | X | X | | | | | | | | | | | | | | | |
| AR Remote Read | | | | | | | | | | | X | | X | | X | | | | | |
| AR Remote Write | | | | | | | | | | | X | | X | | X | | | | | |
| ASE Variable | | | | | | | | | | | | | | | | | | | | |
| Read | | X | X | | X | | | | | | X | X | | X | | | | | | |
| Read List | | X | X | | X | | | | | | | | | | | | | | | |
| Write | | X | X | | X | | | | | | | | | | | | | | | |
| Write List | | X | X | | X | | | | | | | | | | | | | | | |
| Information Report | X | | | | | | X | | X | | | | | | | | X | | X | |
| Information Report List | X | | | | | | X | | X | | | | | | | | X | | X | |
| Exchange | | X | X | | X | | | | | | | | | | | | | | | |
| ExchangeList | | X | X | | X | | | | | | | | | | | | | | | |
| ASE Event | | | | | | | | | | | | | | | | | | | | |
| Unconfirmed Ack Events | X | | | | | | X | | X | | | | | | | | X | | X | |
| Confirmed Ack Event | | X | X | | X | | | | | | | | | | | | | | | |
| Ack Event List | | X | X | | X | | | | | | | | | | | | | | | |

| | Unc | Cnf | Client | Serveur | Éditeur push | Abonné push | Gesti. Édition Pull | Editeur pull | Abonné pull | Report Src | Report Sink |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Services de FAL** | | | **req rcv** | **req rcv** | **req rcv** | **req rcv** | **req rcv** | **req rcv** | **req rcv** | **req rcv** | **req rcv** |
| Enable Event | | X | X | X | | | | | | | |
| Enable Event List | | X | X | X | | | | | | | |
| Get Event Summary | | X | X | X | | | | | | | |
| Get Event Summary List | | X | X | X | | | | | | | |
| Query Event Summary List | | X | X | X | | | | | | | |
| Event Notification | X | | | | X | X | | | | X | X |
| Notification Recovery | X | | | | X | X | | | | X | X |
| ASE Load Region | | | | | | | | | | | |
| Initiate Load | | X | X | X | | | | | | | |
| Push Segment | | X | X | X | | | | | | | |
| Pull Segment | | X | X | X | | | | | | | |
| Terminate Load | | X | X | X | | | | | | | |
| Discard | | X | X | X | | | | | | | |
| ASE Function Invocation | | | | | | | | | | | |
| Start | | X | X | X | | | | | | | |
| Stop | | X | X | X | | | | | | | |
| Resume | | X | X | X | | | | | | | |
| Reset | | X | X | X | | | | | | | |
| Kill | | X | X | X | | | | | | | |
| Action Invoke | X | | | | | | | | | X | X |
| Action Return | X | | | | | | | | | X | X |

# 7 Spécification du modèle de communications de Type 5

## 7.1 Concepts

### 7.1.1 Vue d'ensemble

Le présent paragraphe introduit le concept d'entité application de la FAL (FAL application entity (FAL AE ou bien AE de la FAL)). Il est appelé Agent Field Device Access (accès à l'appareil de terrain (FDA)). L'agent FDA se compose d'ASE APO et AR spécifiques au Type 5 et d'un sous-ensemble d'ASE APO du Type 9. Un sous-ensemble des types de données (data type) définis à l'Article 4 est pris en charge. Les ASE, classes et services du Type 9 pris en charge par le Type 5 sont définis en 14.3.

L'Agent FDA mappe l'ASE de Type 5 et de Type 9 sur un ensemble de points d'extrémité de service de communications en mode orienté connexion et en mode sans connexion sous-jacents modélisés comme des sockets (c'est-à-dire des ports de connexion) Ces services sont appelés services socket dans le reste du présent article.

Les services socket sont indépendants de protocoles réels utilisés pour fournir leurs services. Tout protocole qui peut être mis en correspondance avec ces services peut être utilisé. Les sockets sont adressés par adresse réseau et numéro de port.

Le Type 5 prend en charge l'accès application aux appareils de terrain tant du Type 9 que du Type 5. L'accès à des appareils de terrain du Type 9 est fourni par une classe d'appareil s de Type 5 appelés *Appareils de liaison* (Linking Device). Les appareils de liaison contiennent un Agent FDA qui fournit une fonction passerelle entre les réseaux du Type 5 et du Type 9.

Une autre classe d'appareils de Type 5 est capable d'interconnecter un réseau de Type 5 à des liens qui prennent en charge d'autres types de couches liaison de données. Ces appareils du Type 5 s'appellent des appareils passerelles (Gateway Device). L'Agent FDA dans les

appareils passerelles fournit des services de passerelles qui mettent en correspondance des APDU de Type 5 avec leurs contreparties sur d'autres liaisons.

Une autre classe d'appareils de Type 5 est capable de prendre en charge des connexions LAN redondantes. Ces appareils de Type 5 sont appelés des LAN Redundancy Capable Devices (Appareils capables de redondance LAN). L'Agent FDA dans ces appareils fournit des services LAN Redundancy ASE (ASE de redondance LAN) à l'entité de redondance LAN (LAN Redundancy) de l'appareil.

L'Agent FDA utilise une base de données MIB de Type 5 pour stocker les données. La spécification de cette MIB ne relève pas du domaine application de la présente norme.

Dans tout le reste du présent article, les appareils de Type 5 sont des appareils de la FAL qui contiennent un Agent FDA, alors que les appareils de Type 9 se réfèrent à des appareils qui mettent en œuvre la FAL du Type 9 FAL et la DLL du Type 1.

### 7.1.2 Objectifs de l'agent FDA

L'objectif principal de l'Agent FDA est d'acheminer des services ASE de Type 5 et Type 9 en utilisant des services socket. Cela permet que des appareils de terrain de Type 5 et de Type 9, des appareils E/S conventionnels et d'autres appareils E/S soient connectés par l'intermédiaire d'un Agent FDA dans un *Appareil de liaison* ou *Appareil passerelle*.

Un autre objectif de l'Agent FDA est de rééditer les données de Type 9 d'une liaison de Type 9 vers une autre dans des appareils de Type 9 ayant plusieurs ports qui ne prennent pas en charge le pontage des couches de liaison de données de Type 9. Cela permet à un agent FDA dans un appareil de liaison d'intégrer plusieurs interfaces autonomes de Type 9 au lieu d'utiliser un pont de Type 9.

Un autre objectif de l'agent FDA est d'envoyer et de recevoir des messages pour prendre en charge des applications chargées d'ajouter et de supprimer des objets application de Type 5 dans le réseau et de localiser des objets application sur le réseau.

Un autre objectif de l'agent FDA est d'envoyer et de recevoir des messages de diagnostic pour prendre en charge des applications de redondance LAN.

Par conséquent, l'agent FDA permet:

- la construction de systèmes de commande à partir d'appareils de Type 5 et de Type 9,
- la liaison de réseaux de Type 5 et de Type 9 par des appareils de liaison, et
- l'accès par des applications distantes d'appareils de terrain de tout type par le biais de services de socket utilisant le modèle de communications de Type 5.

### 7.1.3 ASE Data type

#### 7.1.3.1 Types de données

Les data types pris en charge sont un sous-ensemble de ceux définis à l'Article 4. Ce sont:

- Boolean
- Integer8
- Integer16
- Integer32
- Unsigned8
- Unsigned16

- Unsigned32
- Floating Point ("Virgule flottante")
- VisibleString
- OctetString
- Date
- TimeOfDay
- TimeDifference
- BitString
- TimeValue

#### 7.1.3.2 Niveau d'imbrication

Un seul niveau d'imbrication est pris en charge. À savoir, les types construits ne peuvent pas contenir de types construits.

### 7.1.4 ASE d'APO

#### 7.1.4.1 Vue d'ensemble

L'agent FDA prend en charge les ASE APO de Type 5 et de Type 9 tels que spécifiés en 14.3. Les ASE de Type 5 sont les ASE VFD, SMK et LAN Redundancy. L'ASE VFD de Type 5 est une adaptation de l'ASE VFD de Type 9. L'agent FDA prend également en charge les versions des AR de type 9 QUB-CO et QUU qui sont adaptées à être utilisées sur des sockets.

#### 7.1.4.2 ASE LAN Redundancy

L'ASE LAN Redundancy spécifie les services qui prennent en charge les entités LAN Redundancy. Une "LAN Redundancy Entity» dans les appareils capables de redondance LAN utilise les services de l'agent FDA pour envoyer et recevoir des APDU ASE de redondance LAN. Ces APDU sont utilisées par la LAN Redundancy Entity pour construire une table de statuts de réseau afin de choisir des chemins qui évitent les pannes de réseau.

#### 7.1.4.3 ASE System management kernel (SMK)

##### 7.1.4.3.1 SMK

L'entité SMK est l'entité dans un appareil qui est chargée du fonctionnement de la gestion système d'un appareil. L'ASE SMK définit un ensemble de services qui prennent en charge l'acheminement des demandes et des réponses de gestion système à destination et en provenance d'entités SMK.

Lorsqu'ils résident dans un appareil de Type 5, les services SMK sont acheminés par des AR de Type 5. Lorsqu'ils résident dans un appareil d'un autre type, les services SMK peuvent être acheminés par accès direct à des services en mode sans connexion de liaison de données.

Comme les services d'ASE de Type 9, les services d'ASE SMK peut être acheminés entre un appareil connecté à un réseau de Type 5 et un appareil connecté à un réseau qui utilise une couche liaison de données sous-jacente de Type 9. Dans ce cas, l'agent FDA dans un appareil de liaison reçoit une demande ou réponse de service SMK issue d'un réseau et assure le mapping pour un acheminement par l'autre.

##### 7.1.4.3.2 Services de l'ASE SMK

Un service d'ASE SMK est défini pour identifier des appareils.

Deux services d'ASE SMK sont définis pour ajouter et supprimer des appareils de Type 5 et des appareils de Type 9 connectés à un appareil de liaison.

– L'un de ces services est utilisé par le SMK dans un nouvel appareil pour annoncer sa présence périodiquement sur le réseau.
– L'autre service est utilisé par une application de configuration recevant l'annonce de retourner des informations de configuration de base vers le SMK. Cela permet au SMK d'apprendre son rôle dans le système et d'initialiser les opérations normales.

Deux services d'ASE SMK supplémentaires plus sont définis pour trouver les appareils et objets nommés sur le réseau.

#### 7.1.4.4 ASE Virtual field device (VFD)

##### 7.1.4.4.1 Objet VFD

L'objet VFD de Type 9 est utilisé dans le Type 5. Dans le Type 5, tous les objets et services définis par les ASE de Type 9 sont utilisés pour accéder à des APO aussi d'un VFD, avec les exceptions suivantes. La VCR de Type 5 remplace la VCR de Type 9 alors que le service Initiate de Type 5 remplace le service Initiate de Type 9. Tous les services de VFD de Type 5 et de Type 9 sont acheminés sur des réseaux de Type 5 en utilisant la syntaxe et les procédures de Type 5 définies dans la CEI 61158-6-5.

##### 7.1.4.4.2 Relations de communication virtuelles (Virtual communication relationships (VCR))

###### 7.1.4.4.2.1 Vue d'ensemble

Les VCR de Type 5 sont similaires aux VCR de Type 9 (définie par l'ASE Context Management de Type 9). Les VCR de Type 5 fournissent une interface de services du Type 9 pour un accès tant aux VFD du Type 5 qu'aux VFD de Type 9. Les VFD du Type 9 sont accessibles par l'intermédiaire des AE de Type 5 dans des Appareils de liaison. Trois types de VCR sont définis, à savoir Client/Server (Client/Serveur), Publisher/Subscriber (Éditeur/Abonné) et Report Distribution (Distribution de rapports).

###### 7.1.4.4.2.2 VCR de type client/serveur

L'agent FDA prend en charge de façon dynamique les VCR de type client/serveur, comme suit. Les clients envoient des demandes d'initialisation des VCR au serveur FAL AE. Le serveur FAL AE crée de façon dynamique une nouvelle VCR "serveur" et lui délivre la demande pour traitement.

###### 7.1.4.4.2.3 VCR de type éditeur/abonné

Chaque point d'extrémité VCR "abonné" est configuré pour recevoir d'une AR "abonné spécifique. Chaque point d'extrémité VCR de type abonné est configuré pour recevoir ses messages d'un seul point d'extrémité VCR de type éditeur. L'éditeur original peut exister sur une liaison de Type 9 lorsqu'un Appareil de liaison est utilisé pour rééditer les données sur le réseau du Type 5. En plus, si l'abonné est localisé dans un Appareil de liaison, il peut être configuré pour rééditer les données sur une liaison du Type 9.

Lorsque les données sont acheminées par un Appareil de liaison, l'AE FAL met en correspondance les VCR "éditeur/abonné" du Type 9 aux VCR "éditeur/abonné" du Type 5, recevant sur l'une les données produites et les rééditant sur l'autre.

L'agent FDA peut être configuré pour trois types d'édition:

– Une application de Type 5 locale peut éditer des données sur le réseau de Type 5 par l'intermédiaire de l'agent FDA.

– L'agent FDA dans un Appareil de liaison peut rééditer des données d'un Appareil de Type 9 vers le réseau de Type 5.

– L'agent FDA dans un Appareil de liaison peut rééditer des données d'un Appareil de Type 5 vers une liaison de Type 9.

#### 7.1.4.4.2.4 VCR de type "distribution de rapports"

Les VCR de type Report Distribution fonctionnent d'une façon analogue à celle des VCR de type "publisher/subscriber".

L'agent FDA peut être configuré pour trois types de distribution de rapports:

– une application de Type 5 locale peut envoyer des rapports sur le réseau de Type 5 par l'intermédiaire de l'agent FDA.

– l'agent FDA d'un appareil de liaison transmet des rapports reçus en provenance d'appareils sur une liaison de Type 9 vers le réseau de Type 5;

– l'agent FDA d'un appareil de liaison peut transmettre des rapports reçus en provenance d'un appareil de Type 5 vers une liaison de Type 9.

#### 7.1.4.4.3 Service Initiate

Le service Initiate peut être dirigé vers l'adresse générique d'un VFD. Cela permet à plusieurs applications clientes d'ouvrir un accès simultané au VFD en utilisant la même adresse. L'ASE VFD fonctionnant à l'adresse en question crée dynamiquement une nouvelle VCR pour traiter chaque demande de service Initiate reçue.

Le service Initiate aussi peut être dirigé vers le VFD de Base de données MIB (MIB) VFD d'un appareil sans connaître son adresse générique. Seule l'adresse de l'appareil est nécessaire pour ouvrir l'AR de l'appareil.

Le service Initiate fonctionne localement pour les VCR éditeur/abonné. Il ouvre la VCR pour l'édition ou pour l'abonnement.

### 7.1.5 Relations entre applications

#### 7.1.5.1 Transferts sans session

L'agent FDA envoie et reçoit les messages de SM et de LAN Redundancy sans sessions FDA en utilisant des services sans connexion sous-jacents. Les transferts sans session ne sont pas capables de concaténer des messages.

#### 7.1.5.2 Sessions FDA et VCR de Type 5

Les VCR de Type 5 fournissent l'accès à des VFD à travers le Type 5. Elles sont les contreparties de Type 5 des VCR de Type 9. Les VCR de Type 5 utilisent des sessions FDA pour transférer des services de Type 9.

Les sessions FDA fournissent des communications à destination/en provenance d'agents FDA. Elles sont les contreparties de Type 5 des AR de Type 9. Elles sont établies pour la durée de vie de la session pour fonctionner soit sur connexion TCP, soit sur des services sans connexion, entre des ports source et destination.

Les similitudes entre les AR de Type 9 et les sessions FDA sont les suivantes.

– Les AR de Type 9 et les sessions FDA prennent en charge les mêmes modèles de VCR, à savoir client/serveur, éditeur/abonné et Distribution de rapports.

– Les appareils de Type 9 et les agents FDA peuvent chacun contenir une ou plusieurs sessions de points d'extrémité AR/FDA du Type 9.

– Les AR de Type 9 sont identifiées sur le câble par les DLCEP/DLSAP source et destination. Les sessions FDA sont identifiées sur le câble par les adresses de réseau source et destination (adresse réseau plus numéro de port).

– Les SMK de Type 9 communiquent directement avec la liaison de données en utilisant le service Unitdata. Les messages SM de FDA sont acheminés d'une manière similaire, avec l'aide de services sans connexion, et ce, sans utiliser une session.

NOTE Les AR de Type 9 et les sessions FDA sont négociées pour fonctionner en utilisant une version spécifique de leurs unités de données de protocole ((PDU)/messages. Toutes les PDU/ tous les messages envoyé(e)s sur une AR de Type 9/une session utilisent la même version de protocole. Les agents FDA qui prennent en charge des versions plus récentes prennent également en charge des versions antérieures pour des raisons de compatibilité ascendante.

– Les AR de Type 9 et les sessions FDA diffèrent en ce qui suit.

– Une même session est capable de prendre en charge plus d'une VCR de Type 5. Les AR de Type 9 ne sont capables de prendre en charge qu'une seule VCR de Type 9.

– Les sessions sont capables de concaténer plusieurs messages en une seule et même PDU. Chaque PDU de DLL n'achemine qu'un seul message (PDU) de Type 9.

– Les sessions client/serveur sans connexion de Type 5 sont établies en utilisant des services sans connexion sous-jacents. Pour les AR de Type 9, il n'existe pas de concept d'AR client/serveur sans connexion.

– Les sessions client/serveur orientées connexion sont établies sur TCP. Une fois qu'une connexion TCP a été établie, le client envoie un message FDA de demande de service Open Session sur la connexion en question pour ouvrir la session. Les AR de Type 9 opèrent de la même façon, mais la connexion est établie au niveau de la couche liaison de données et la demande de connexion de liaison de données contient la demande FAS d'ouverture de l'AR.

#### 7.1.5.2.1 Client/serveur

**Sessions client/serveur**

Les sessions client/serveur sont utilisées pour prendre en charge les VCR client/serveur. Une même session client/serveur peut prendre en charge plus d'une VCR. Il convient d'établir les sessions client/serveur avant qu'elles ne puissent être utilisées. Elles sont établies par l'échange de messages FDA de demande de service Open Session et de réponse entre le client et le serveur.

Si le protocole TCP est à utiliser, le client envoie d'abord une demande de connexion TCP vers le port FDA réservé de l'appareil serveur. L'agent FDA au niveau du serveur se met à l'écoute, sur le port FDA réservé, des demandes de connexion TCP et lorsqu'il en reçoit une, il créait une nouvelle session pour le serveur du point d'extrémité. Il attend ensuite la prochaine demande de connexion TCP, alors que la session du point d'extrémité nouvellement créée attend une demande de service Open Session.

Si des services sans connexion sous-jacents sont à utiliser, le client envoie le message de demande de service Open Session au port de Type 9 réservé de l'appareil serveur. L'agent FDA au niveau du serveur se met à l'écoute, sur le port FDA réservé, des demandes de service Open Session et lorsqu'il en reçoit une, il créait une nouvelle session serveur pour le point d'extrémité, lui alloue un socket sans connexion avec une nouvelle adresse de port et lui livre la demande Open Session. Si la création d'une nouvelle session serveur pour le point d'extrémité n'est pas possible, une réponse négative est envoyée à partir du port FDA. L'agent FDA attend ensuite la prochaine demande de service Open Session.

Lorsqu'une nouvelle session serveur du point d'extrémité reçoit la demande de service Open Session, il traite le message. Le message FDA de demande de service Open Session contient des paramètres négociables qui permettent au serveur de configurer dynamiquement la session serveur du point d'extrémité et qui indiquent si, oui ou non, la session sera utilisée comme session de *configuration* (voir 7.1.5.4).

Si la demande est acceptée, la session du point d'extrémité nouvellement créée retourne une réponse positive. Si la demande est rejetée, la session du point d'extrémité nouvellement créée retourne une réponse négative, ferme le socket sous-jacent et s'auto supprime. Les réponses sont retournées à partir du socket sous-jacent, ou à partir du port de Type 9 réservé s'il n'était pas possible d'ouvrir un nouveau socket sans connexion.

Les sessions client/serveur sont arrêtées par une demande de l'utilisateur, par l'absence d'activité ou à cause de problèmes liés au protocole FDA.

**VCR de type client/serveur**

Les VCR client/serveur sont utilisées pour acheminer des demandes et des réponses à destination et en provenance des VFD. Les VCR client/serveur peuvent être établies entre une application cliente et un VFD soit de Type 5, soit de Type 9. Les VFD de Type 9 sont accessibles en établissant une VCR de Type 5 à un appareil de liaison. Cette point d'extrémité VCR de Type 5 accède à une VCR de Type 9 qui se connecte au VFD de Type 9. Plusieurs VCR de Type 5, sur des sessions identiques ou différentes, peuvent accéder simultanément à une même VCR de Type 9. Les appareils de terrain de Type 9 n'ont donc pas nécessairement à fournir des points d'extrémité VCR de type serveur individuel pour prendre en charge chaque client concomitant de Type 5.

Il convient d'établir les VCR client/serveur avant qu'elles ne puissent être utilisées. Les VCR client/serveur sont établies par l'échange de message de demande et de réponse de service Initiate entre le client et le serveur après l'ouverture d'une session entre eux.

Le client envoie le message de demande de service Initiate vers le point d'extrémité VCR *générique* pour le VFD serveur. Le point d'extrémité VCR est identifié dans la demande de service Initiate par un sélecteur *générique* ou par une valeur spéciale qui indique *NMA Access*. La valeur du sélecteur générique est contenue dans la liste des VFD dans la MIB pour l'appareil. Elle peut être obtenue en utilisant le service Find Tag Query pour l'indicateur du VFD ou pour un objet contenu dans le VFD.

Le point d'extrémité VCR générique se met à l'écoute, de ce sélecteur, des demandes de service Initiate et lorsqu'il en reçoit une, il crée un nouveau point d'extrémité VCR de type serveur et ajoute une entrée dans la liste MIB des VCR *automatiques*. Le point d'extrémité VCR générique livre aussi la demande de service Initiate au nouveau point d'extrémité VCR et attend ensuite la prochaine demande de service Initiate.

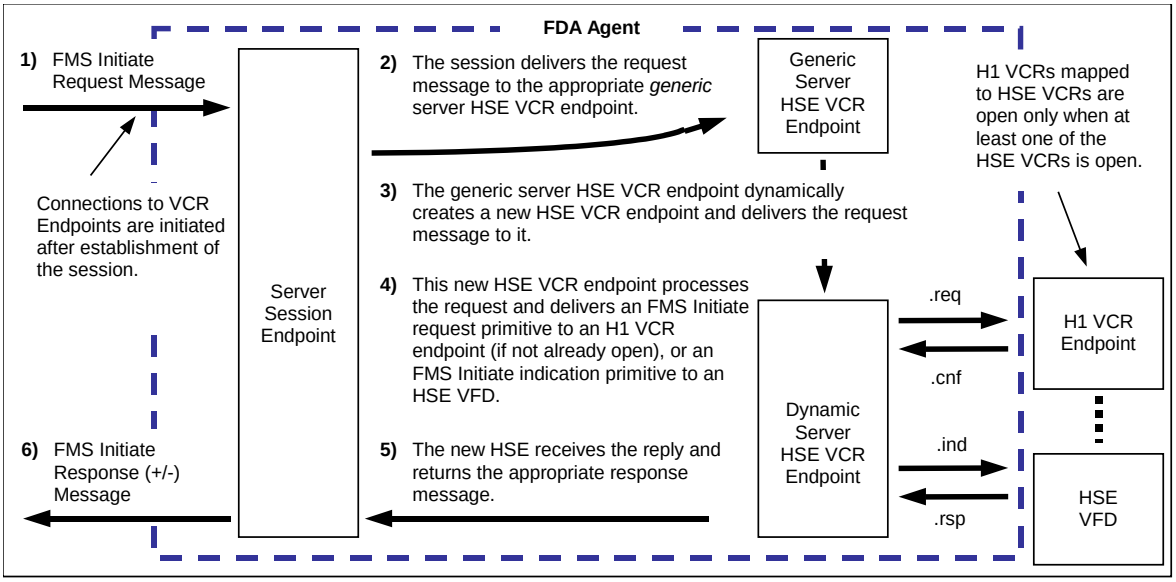
Le nouveau point d'extrémité VCR utilise des informations d'adresse contenues dans la demande pour déterminer si, oui ou non, un VFD est un VFD de type 5 local ou, dans le cas d'un appareil de liaison, si, oui ou non, il s'agit d'un VFD de Type 9. S'il s'agit d'un VFD de Type 5 local ou s'il s'agit d'un VFD d'agent de gestion de Type 9 dans l'appareil de liaison, le point d'extrémité VCR produit une primitive "indication" de service Initiate et attend la réponse. S'il s'agit d'un VFD de Type 9 situé dans un appareil de Type 9 relié à l'appareil de liaison, le point d'extrémité de VCR présente une primitive "request" de service Initiate au point d'extrémité VCR de Type 9 client si ce dernier n'est pas déjà ouvert. Si le point d'extrémité VCR de Type 9 est ouvert, le point d'extrémité VCR de Type 5 local s'y attache localement.

Si la demande est acceptée par le VFD local ou si la VCR de Type 9 s'ouvre ou est déjà ouverte, le point d'extrémité VCR nouvellement créé retourne une réponse positive qui contient l'indice (Index) de l'OD pour sa nouvelle entrée dans la liste de MIB des VCR *automatiques*. Cette valeur est utilisée pour identifier le point d'extrémité VCR dans tous les messages envoyés sur la VCR. Si la demande a été envoyée en utilisant l'accès NMA et la session sous-jacente n'est pas une session de configuration, le point d'extrémité VCR rejette toutes demandes de mise à jour qu'elle reçoit du client.

Si la demande de service Initiate est rejetée, le point d'extrémité VCR nouvellement créé retourne une réponse négative, retire son entrée de la liste MIB des VCR *automatiques* et s'auto supprime.

Les VCR client/serveur sont arrêtées des mêmes façons qu'il est mis fin aux VCR de Type 9. En outre, elles sont arrêtées lorsqu'elles n'ont plus d'activité qui s'y déroule. Les états et le traitement associés au déclenchement et à l'arrêt des VCR sont décrits dans la CEI 61158-6-5.

La Figure 9 montre l'utilisation d'une session pour ouvrir des VCR de Type 5 et de Type 9.




**Légende**

| Anglais | Français |
|---|---|
| FDA Agent | Agent FDA |
| FMS Initiate Request Message | Message de demande de lancement FMS |
| The session delivers the request message to the appropriate *generic* server HSE VCR endpoint. | La session livre le message de demande au serveur HSE VCR du point d'extrémité adéquat |
| Generic Server HSE VCR Endpoint | serveur générique PHSE VCR du point d'extrémité |
| H1 VCRs mapped to HSE VCRs are open only when at least one of the HSE VCRs is open. | Les VCR H1 mappées à des VCR HSE sont ouvertes que si au moins l'une VCR HSE est ouverte. |
| The generic server HSE VCR endpoint dynamically creates a new HSE VCR endpoint and delivers the request message to it. | Le serveur générique VCR HSE du point d'extrémité crée dynamiquement un nouveau point d'extrémité CR HSE et lui livre le message de demande |
| Connections to VCR Endpoints are initiated after establishment of the session | Les connexions aux points d'extrémité VCR sont lancées après l'établissement de la session. |
| Server Session Endpoint | Point d'extrémité de la session serveur |
| This new HSE VCR endpoint processes the request and delivers an FMS Initiate request primitive to an H1 VCR endpoint (if not already open), or an FMS Initiate indication primitive to an HSE VFD | Ce nouveau point d'extrémité VCR HSE traite la demande et délivre une primitive «request» de service Initiate FMS à un point d'extrémité de VCR H1 (si pas encore ouvert) ou une primitive «indication» de service Initiate FMS à un VFD HSE. |
| Dynamic Server HSE VCR Endpoint | Point d'extrémité VCR HSE serveur dynamique |
| H1 VCR Endpoint | Point d'extrémité VCR H1 |

| Anglais | Français |
|---|---|
| FMS Initiate Response (+/-) Message | Message de réponse (+/-) au service Iniitiate FMS |
| The new HSE receives the reply and returns the appropriate response message. | Le nouveau HSE reçoit la réponse et retourne le message de réponse approprié. |
| HSE VFD | HSE VFD |

**Figure 9 – Déclenchement de VCR**

#### 7.1.5.2.1.1 Maintien à l'état ouvert des VCR client/serveur et de leurs sessions

Les sessions et VCR client/serveur se ferment automatiquement lorsqu'il n'y a aucun trafic de message pendant une durée appelée Inactivity Close Time (Durée de fermeture pour cause d'inactivité). Pour prévenir l'arrêt des VCR à cause d'une inactivité, le point d'extrémité VCR de type client envoie un message de demande de service Idle confirmé lorsque son temporisateur Idle expire. Son temporisateur Idle est relancé en utilisant au maximum le tiers de sa durée de fermeture pour cause d'inactivité chaque fois qu'il envoie un message. Le point d'extrémité VCR de type serveur qui reçoit le message de demande de service Idle retourne immédiatement le message de réponse de service Idle correspondant. Lorsqu'il n'y a aucune VCR active sur une session, aucun message Idle n'est envoyé. Cela fait expirer la session et la fait se fermer.

### 7.1.5.2.2 Éditeur/abonné

#### 7.1.5.2.2.1 Généralités

Les sessions éditeur/abonné ont été adaptées de leurs contreparties AR de Type 9. Les sessions éditeur/abonné fonctionnent comme suit.

– Les messages ne sont pas envoyés par des points d'extrémité éditeur ou abonné pour établir des sessions, contrairement aux connexions éditeur/abonné de Type 9.
– Les données produites sont envoyées en multidiffusion en utilisant des services sans connexion sous-jacents vers une adresse de multidiffusion configurée et vers un numéro de port configuré.
– Les éditeurs n'utilisent que des messages de service confirmé pour éditer des données.
– Chaque appareil peut être configuré avec zéro, une ou plusieurs sessions pour le point d'extrémité de type éditeur utilisées pour émettre des données produites.
– Les abonnés ne sont pas connus de l'éditeur.
– Il peut y avoir zéro, un ou plusieurs abonnés recevant des données produites.
– Chaque appareil peut être configuré avec zéro, une ou plusieurs sessions pour le point d'extrémité de type abonné utilisées pour recevoir des données produites.

Les VCR éditeur/abonné ont été adaptées de leurs contreparties VCR de Type 9. Les VCR éditeur/abonné fonctionnent comme suit.

– Les messages ne sont pas envoyés par les points d'extrémité éditeur ou abonné pour établir des VCR.
– La détection de messages perdus, de messages dupliqués et de messages en retard est possible, bien que la récupération des messages perdus ne soit pas possible. Cependant, des applications peuvent avoir leurs propres méthodes de récupération d'informations perdues (en accédant à des rapports de tendance, par exemple).
– Zéro, une ou plusieurs points d'extrémité de VCR de type éditeur peuvent utiliser le même point d'extrémité de session de type éditeur pour éditer leurs données.
– Un point d'extrémité de VCR de type éditeur peut être configuré pour éditer des données pour un FBAP (Function Block Application Process, c'est-à-dire «Processus application de bloc fonctionnel).

- Un point d'extrémité de VCR de type éditeur peut être configuré pour rééditer des données reçues par un point d'extrémité de VCR de type abonné de Type 9.
- Chaque point d'extrémité de VCR de type abonné reçoit des données issues d'un seul point d'extrémité de VCR de type éditeur.
- Un point d'extrémité de VCR de type abonné peut être configuré pour livrer, à une application locale, des données reçues. Un point d'extrémité de VCR de type abonné distinct est requis pour chaque application locale qui reçoit les données produites.
- Un point d'extrémité de VCR de type abonné peut être configuré pour livrer des données reçues par un point d'extrémité VCR de type éditeur de Type 9, amenant celle-ci à rééditer des données de Type 5 sur une liaison de Type 9. Un point d'extrémité VCR de type abonné distinct est requis pour chaque liaison de Type 9 qui est destinée à rééditer les données.

#### 7.1.5.2.2.2 Établissement de session et de VCR

Les points d'extrémité éditeur/abonné sont établis localement. Aucun message FDA n'est échangé pour les établir.

#### 7.1.5.2.2.3 Cohérence temporelle

Les applications présentent des données produites à la VCR configurée dans leurs objets de liaison. La VCR les présente à sa session de type éditeur. La session de type éditeur les met en buffer pendant un temps pouvant atteindre au maximum la durée Transmit Delay Time (Durée de retard d'émission) définie pour la session avant que présentation aux points d'extrémité de type abonné ne livre les données reçues vers les applications consommatrices ou vers le Type 9 pour réédition (dans des appareils de liaison) immédiatement à la réception. La cohérence temporelle des données produites dépend donc de la durée Transmit Delay Time éditeur, de la latence du réseau et du retard imposé par les services sans connexion sous-jacents et de l'agent FDA au sein de l'appareil de réception.

La fin de message FDA (FDA Message Trailer) contient également un marqueur temporel facultatif qui peut être livré à des applications consommatrices afin de leur permettre de déterminer si, oui ou non, les données avaient été produites avec une fenêtre temporelle acceptable. Ce marqueur temporel est appliqué au message par le point d'extrémité de session au moment où il présente le message en vue de son transfert. L'utilisation de ce marqueur temporel est configurable.

Le marqueur temporel est transmis entre l'agent FDA et l'application éditeur/abonné par des moyens locaux (qui ne font pas partie de l'interface de services).

### 7.1.5.2.3 Distribution de rapports

Les sessions de type Report Distribution sont identiques aux sessions de type éditeur/abonné, avec les exceptions suivantes.

- Les sources des rapports correspondent aux éditeurs.
- Les puits des rapports correspondent aux abonnés.
- Les rapports sont identifiés par l'adresse FDA de l'en-tête de message FDA et le paramètre *index* dans la partie FDA du message.
- Chaque session puits du point d'extrémité livre à toutes les VCR qui lui sont associées tous les messages reçus.

### 7.1.5.2.4 Arrêt de sessions

Il est mis fin aux sessions pour les raisons suivantes:

- Les sessions client/serveur sont arrêtées en raison d'une inactivité. L'inactivité est définie comme étant l'échec d'une session dans l'état Open à recevoir un message

valide dans les limites de la durée spécifiée par l'attribut de session Inactivity Close Time.

– Les sessions client/serveur sont arrêtées en raison de l'arrêt de la connexion TCP sous-jacente.

– Les sessions client/serveur sont arrêtées en raison la réception d'un message non valide ou d'un second message de demande de service Open Session. Ces messages sont une indication que le flux de données a perdu la synchronisation ou communique avec un partenaire non conforme. La question de savoir si, oui ou non, des sessions de type abonné du point d'extrémité ferment en raison de la réception d'un message non valide est dépendante de la mise en œuvre.

– Demande d'utilisateur. Les demandes d'utilisateur pour fermer une session sont dépendantes de la mise en œuvre.

Lorsqu'il est mis fin à une session en raison d'une inactivité, en raison d'une demande d'utilisateur ou en raison d'un message non valide ou d'un second message de demande de service Open Session, l'agent FDA exécute les étapes ci-après, et ce, dans l'ordre. Lorsqu'il met fin à une session parce que la connexion TCP sous-jacente a été arrêtée, l'agent FDA n'exécute que les étapes 1, 4 et 5.

1) Arrête d'accepter des demandes d'envoyer des messages sur la session,
2) Envoie des messages Abort de Type 9 sur la totalité de ses VCR de Type 5 ouvertes,
3) Met fin aux connexions TCP support, le cas échéant.
4) Livre toutes les primitives de confirmation et d'erreur mises en files d'attente et rejette toutes les primitives "indication" mises en file d'attente,
5) Ferme les points d'extrémité VCR correspondantes et émet une primitive "indication" de service Abort de Type 9 pour chacune.

NOTE Des applications peuvent fermer des VCR en utilisant le service Abort de Type 9 sans fermer la session FDA de support.

#### 7.1.5.3 Initialisation d'agent FDA

Lorsque l'agent FDA démarre, il initialise ses sessions de type serveur du point d'extrémité en utilisant les valeurs par défaut des attributs. Une fois qu'elle est initialisée, elle est prête à envoyer et à recevoir des messages pour le SMK et pour l'entité LAN Redundancy. Une fois que le SMK a atteint son état opérationnel, il permet à l'agent FDA d'envoyer et de recevoir des messages de toutes les autres applications d'appareils/tous les autres VFD.

#### 7.1.5.4 Configuration des sessions et des VCR

Avant que l'appareil ne puisse devenir complètement opérationnel, il est nécessaire de configurer sa base de données MIB. Les informations relatives aux sessions FDA et aux VCR sont contenues dans la MIB.

La configuration de la MIB est accomplie en utilisant des sessions de *configuration* de type client/serveur.

Seules les sessions de configuration de NMA sont autorisées à modifier les informations dans la MIB, dans les MIB d'interface de Type 9 d'appareil de liaison et dans les MIB de Type 9 d'appareil de Type 9 accessibles par le biais de l'appareil de liaison. Il n'est permis d'ouvrir qu'une seule session de configuration à la fois pour un appareil de Type 5. Après que la session de configuration ait été ouverte, des VCR distinctes sont ouvertes à l'agent de gestion de Type 5 et à chaque agent de gestion de Type 9 en utilisant le service Initiate avec l'option de connexion NMA Access (accès NMA).

Les sessions de configuration fournissent également aux applications de configuration l'accès à des VFD autres que les VFD NMA. Cependant, contrairement aux VFD NMA, d'autres sessions peuvent écrire simultanément dans ces VFD autres que NMA.

Les sessions de type client/serveur autres que de configuration fournissent un accès en lecture aux VFD NMA après les avoir connectés en utilisant le service Initiate avec l'option de connexion NMA Access. Les sessions de type client/serveur autres que de configuration fournissent un accès en lecture-écriture à tous les autres VFD. La VCR est chargée de filtrer les services de mise à jour présentés pour la mise à jour de la MIB qui utilise ces sessions. La liste des services de mise à jour qui sont filtrés sur ces VCR est la suivante:

– Generic Initiate Download Sequence
– Generic Download Segment
– Generic Terminate Download Sequence
– Initiate Download Sequence
– Download Segment
– Terminate Download
– Request Domain Download
– Write
– Write with Subindex

NOTE Les sessions client/serveur et les VCR client/serveur sont définies par défaut et ne sont pas configurées, sauf si elles servent à prendre en charge l'accès à partir d'un client de Type 9 à travers le réseau de Type 5. Dans ce cas, seuls les points d'extrémité clients sont configurés; le point d'extrémité serveur acquiert dynamiquement ses attributs pendant l'établissement des points d'extrémité, comme il le fait dans tous les autres cas.

Les sessions de type Publisher/Subscriber et Report Distribution Sessions ainsi que leurs VCR associées sont toujours configurées.

#### 7.1.5.5 Acheminement des services

##### 7.1.5.5.1 Généralités

L'agent FDA achemine les services FDA, VFD, SM et LAN Redundancy ainsi que les services de Type 9. Les services VFD et de Type 9 sont utilisés pour accéder à des VFD et à leurs objets. Les services SM sont utilisés pour accéder aux SMK. Les services LAN Redundancy sont utilisés pour prendre en charge les interfaces réseau redondantes. Afin d'acheminer ces services, l'agent FDA:

– reçoit des primitives de services issues d'applications locales et/ou de la pile de Type 9 et les convertit en messages FDA qu'il présente à l'interface socket sous-jacente.
– reçoit des messages issus de son interface socket sous-jacente et les convertit en primitives de services qu'il livre à des applications locales et/ou à la pile de Type 9. Si une session qui n'est pas de configuration reçoit un message de demande de mise à jour pour la MIB, le point d'extrémité VCR éditeur envoie une réponse négative en utilisant une classe d'erreur = "access" et un code d'erreur = "accès objet refusé", au lieu de livrer le message à la MIB.
– reçoit des demandes de services SM et des réponses qu'il présente à l'interface aux services sans connexion sous-jacents.
– reçoit des messages SM issus de son interface aux services sans connexion sous-jacents et les livre comme indication et confirmation de services SM.
– reçoit des demandes de services LAN Redundancy qu'il présente à l'interface aux services sans connexion sous-jacents.
– reçoit des messages LAN Redundancy issus de son interface aux services sans connexion sous-jacents et les livre comme indication et confirmation de LAN Redundancy.

Les paragraphes suivants débattent en détail de ces sujets.

#### 7.1.5.5.2 Transferts de buffers

Les sessions du point d'extrémité mettent en buffer les données selon leur attribut Transmit Delay Time. Cet attribut permet à des points d'extrémité de concaténer des messages dans un buffer et les transférer en une seule et même émission. Voir 7.1.5.7.4, Concaténation de message, pour une description.

#### 7.1.5.5.3 Mise en correspondance des demandes et des réponses

Comme énoncé en 1.2, la présente norme ne traite pas de la méthode par laquelle un protocole associé apparie une réponse à une demande induite. Par conséquent, la description suivante est strictement hypothétique.

Lorsque l'agent FDA reçoit de son socket sous-jacent un message de demande de service confirmé, il le livre ou le transmet au VFD de destination et attend la réponse. Lorsque la réponse est reçue, il la transmet au demandeur.

Un ID d'instance de demande peut être contenu à la fin du message (Message Trailer) pour apparier la réponse reçue issue du VFD à la demande correspondante. Une valeur est placée dans le champ ID d'instance de l'APDU de demande et la même valeur est retournée dans ce champ dans la PDU de réponse. Lorsqu'une demande est en cours (réponse pas encore reçue), il convient que l'AE n'utilise pas le même Instance ID pour une autre demande qu'il envoie au même serveur en utilisant le même protocole identifié dans l'en-tête de message (Type 9 ou SM, par exemple). L'ID en question peut toutefois être utilisé dans des demandes envoyées vers un serveur différent en utilisant un protocole différent.

L'AE demandeuse détermine chaque valeur de l'ID instance. Le domaine d'application de l'Instance ID est spécifique au type de message (voir Tableau 82).

**Tableau 82 – Domaine d'application de l'Invoke Id**

| Type de Messages | L'Instance ID est unique dans le domaine d'application: |
|---|---|
| des messages SM confirmé | du socket demandeur |
| des messages FDA Open Session et Initiate | du socket demandeur |
| des autres messages confirmés | de la VCR associée, identifiée dans le champ FDA Address de l'en-tête de message |

Lorsqu'une demande est reçue par un appareil de liaison et est à transmettre à une liaison attachée de Type 9, il résout les deux problèmes suivants.

En premier lieu, afin d'assurer l'unicité de l'Instance ID sur une VCR de Type 9, l'agent FDA met en correspondance les Instance ID reçus en provenance de clients différents avec les Instance ID uniques de VCR de Type 9. Une fois que cela a été fait, les réponses reçues sur une VCR de Type 9 VCR sont mises en correspondance en retour au client demandeur et l'instance ID émetteur est reçue. La manière dont l'agent FDA met en œuvre ce mapping n'est pas spécifiée par le présent document.

En second lieu, il met prématurément fin à la VCR client/serveur de Type 9 si la réponse n'est pas retournée dans les limites de temps spécifiées dans la MIB. Après avoir mis prématurément fin à la VCR de Type 9, il retourne une réponse au demandeur en utilisant la classe d'erreurs = "service" et le code d'erreur = "response time-out" (Délais de réponse). Il tente ensuite de rétablir la VCR de Type 9. Si cela ne réussit pas, il met prématurément fin à toutes les VCR qui sont associées à la VCR de Type 9.

#### 7.1.5.5.4 Messages dans un ordre incorrect

Des messages dans un ordre incorrect peuvent être reçus lorsque des services sans connexion sous-jacents sont utilisés. Ordre incorrect signifie que des messages reçus consécutivement ne sont pas reçus dans le même ordre suivant lequel ils avaient été émis.

NOTE Les VCR utilisent des numéros de message pour détecter l'ordre incorrect. Les numéros de message sont utilisés pour numéroter séquentiellement chaque message émis par une VCR. L'ordre incorrect n'est pas détecté pour les transferts de message SM ou LAN Redundancy.

Les numéros de message sont un champ facultatif défini dans la fin de message (Message Trailer). Leur présence dans la fin est configurée pour des sessions éditeur/abonné et Report Distribution et elle est négociée pour des sessions client/serveur. Lorsqu'ils sont définis pour la session, la détection d'un ordre incorrect est activée, quel que soit le type de point d'extrémité de session.

Par l'utilisation de numéro de message, les types d'ordre incorrect qui peuvent être détectés sont donnés dans le Tableau 83.

**Tableau 83 – Types d'ordre incorrect détectables par numéros de message**

| Type d'ordre incorrect | Description |
|---|---|
| aucun | le prochain numéro de message de la séquence est reçu |
| messages dupliqués | le même numéro de message est reçu plus d'une fois |
| messages en retard | un numéro de message plus petit est reçu après un numéro de message plus élevé |
| messages manquants | un trou raisonnable (voir NOTE) dans les numéros de message apparaît entre des messages reçus consécutivement |
| perte de synchronisation | un trou déraisonnable dans les numéros de message apparaît entre des messages reçus consécutivement |
| NOTE La ligne de séparation entre raisonnable et déraisonnable est appelée "bande de garde". Un trou dans les numéros des messages reçus qui excède la bande de garde indique qu'il est déraisonnable de penser que le trou représente des messages manquants. Au contraire, il est un indicateur de la perte de synchronisation avec l'expéditeur. Cela peut se produire parce que l'expéditeur a subi une réinitialisation ou que l'expéditeur a une défaillance et un expéditeur de secours a pris la main. La bande de garde est débattue ci-après. | |

Tous les types de messages dans l'ordre incorrect reçus sur des sessions client/serveur sont livrés par l'agent FDA tels qu'ils sont reçus. Il s'agit du même ordre que l'ordre de livraison fourni lorsque les numéros de message ne sont pas utilisés.

Pour ce qui concerne les VCR éditeur/abonné, la livraison des types de messages dans un ordre incorrect est traitée différemment (voir la description dans le Tableau 84).

**Tableau 84 – Livraison des types de messages dans l'ordre incorrect sur des VCR éditeur/abonné**

| Type d'ordre incorrect | Traitement de point d'extrémité de VCR du type abonné en réception |
|---|---|
| aucun | Le message en séquence est livré |
| messages dupliqués | Le message dupliqué est rejeté. |
| messages en retard | Le message en retard est rejeté. |
| messages manquants | Le trou est ignoré et le message reçu est livré. |
| perte de synchronisation | La perte de synchronisation est ignorée et le message reçu est livré. |

Pour tous les types de VCR, des statistiques peuvent être rassemblées pour prendre en charge la gestion de réseau (voir la description dans le Tableau 85).

**Tableau 85 – Statistiques rassemblées par VCR**

| Type d'ordre incorrect | Statistiques rassemblées |
|---|---|
| aucun | Le nombre de messages en séquence reçus |
| messages dupliqués | Le nombre de messages dupliqués reçus |
| messages en retard | Le nombre de messages en retard reçus |
| messages manquants | Le nombre de messages manqués |
| perte de synchronisation | Le nombre de pertes de synchronisation détectées |

Afin de déterminer la taille d'un trou raisonnable dans les numéros de messages reçus, il est utilisé une *bande de garde*. La *bande de garde* fournit une limite inférieure et une limite supérieure pour les numéros de messages reçus. Il s'agit d'un nombre configuré dans la NMIB et sa valeur par défaut est 5.

L'algorithme suivant utilise la bande de garde et la valeur du numéro de message dans le dernier message livré (Last delivered message (LDM)) pour déterminer le type de l'ordre incorrect.

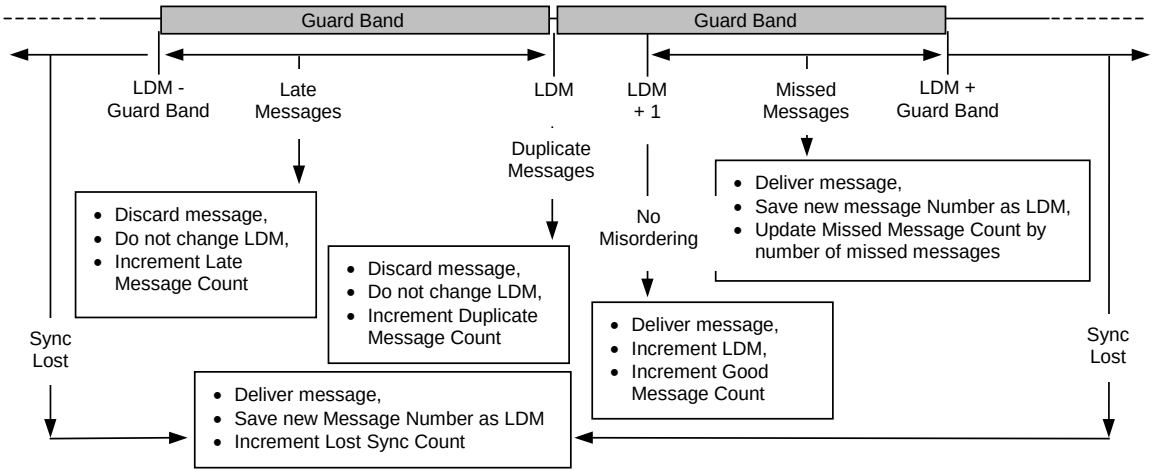
– Lorsqu'un point d'extrémité de VCR de type abonné reçoit son premier message, il le livre et sauvegarde son numéro comme étant la valeur initiale de son LDM.

– À la suite de la réception du premier message, lorsqu'un point d'extrémité de VCR de type abonné reçoit un nouveau message, il détermine le type d'ordre incorrect, en utilisant le module du numéro de message pour tous les calculs (voir Tableau 86).

**Tableau 86 – Détermination du type d'ordre incorrect à une VCR de type abonné**

| Type d'ordre incorrect | Calcul |
|---|---|
| aucun | Numéro de message reçu = LDM + 1 |
| messages dupliqués | Numéro de message reçu = LDM |
| messages en retard | (LDM – bande de garde) =< Numéro de message reçu < LDM |
| messages manquants | LDM + 1 < Numéro de message reçu =< (LDM + bande de garde) |
| perte de synchronisation | (LDM – bande de garde) > Numéro de message reçu OU Numéro de message reçu > (LDM + bande de garde) |

La Figure 10 récapitule le traitement des messages dans l'ordre incorrect.



**Légende**

| Anglais | Français |
|---|---|
| Guard Band | Bande de garde |
| LDM – Guard Band | Bande de garde LDM |
| Late Messages | Messages en retard |
| Duplicate messages | Messages en doublons |
| Missed Messages | Messages manqués |
| Deliver message,<br>Save new message Number as LDM,<br>Update Missed Message Count by number of missed messages | Livrer le message<br>Sauvegarder le nouveau nombre de messages comme LDM<br>Mettre à jour le nombre total de messages manqués par le nombre de messages manqués |
| No Misordering | Aucun mauvais ordonnancement |
| Discard message,<br>Do not change LDM,<br>Increment Late Message Count | Rejeter le message<br>Ne pas changer le LDM,<br>Incrémenter le nombre de messages en retard |
| Discard message,<br>Do not change LDM,<br>Increment Duplicate Message Count | Rejeter le message<br>Ne pas changer le LDM,<br>Incrémenter le nombre de messages en doublons |
| Deliver message,<br>Increment LDM,<br>Increment Good Message Count | Livrer le message<br>Incrémenter LDM<br>Incrémenter le nombre de bons messages |
| Deliver message,<br>Save new Message Number as LDM<br>Increment Lost Sync Count | Livrer le message<br>Sauvegarder le nouveau nombre de messages comme LDM,<br>Incrémenter le nombre de synchronisations perdues |
| Sync Lost | Synchronisation perdue |

**Figure 10 – Traitement des messages dans l'ordre incorrect**

#### 7.1.5.5.5 Conversion des messages en primitives de services

Le Tableau 87 spécifie le mapping du type de message reçu au type de primitive approprié en fonction de l'emplacement du SMK ou VFD auquel l'accès a lieu.

**Tableau 87 – Mapping de messages reçus à des primitives**

| Type de message en entrée | → | Destination finale | Type d'interface FDA | Livré à |
|---|---|---|---|---|
| Demande SM | → | SMK local | Demande SM | SMK local |
| Réponse SM | → | SMK local | Réponse SM | SMK local |
| Demande SM | → | SMK de Type 9 | Primitive "request" | SMK local de Type 5 |
| Message de demande | → | VFD local | Primitive "indication" | VFD local |
| Message de réponse | → | VFD local | Primitive "confirmation" | VFD local |
| Message de demande | → | VFD de Type 9 | Primitive "request" | Interface de la pile communication de Type 9* |
| Message de réponse | → | VFD de Type 9 | Primitive "response" | Interface de la pile communication de Type 9* |
| NOTE Pour les passerelles E/S, l'abstraction de l'interface de la pile communication de Type 9 est maintenue. | | | | |

#### 7.1.5.5.6 Conversion des primitives de services en messages

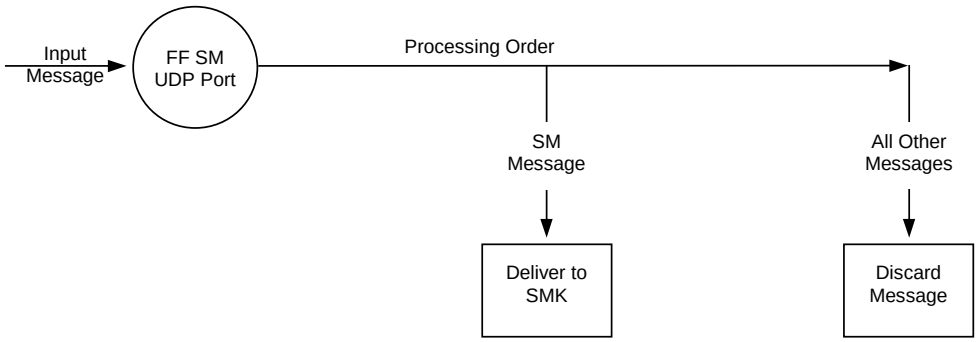
Le Tableau 88 spécifie le mapping du type de primitive reçu au type de message en fonction de l'emplacement du SMK ou VFD émettant la primitive.

**Tableau 88 – Mapping de primitives reçues à des messages**

| Demandeur | Type d'interface FDA | → | Type de message en sortie |
|---|---|---|---|
| SMK local | Primitive "request" | → | Message de demande |
| SMK local | Primitive "response" | → | Message de réponse |
| VFD local | Primitive "request" | → | Message de demande |
| VFD local | Primitive "response" | → | Message de réponse |
| Interface de la pile communication de Type 9* | Primitive "indication" | → | Message de demande |
| Interface de la pile communication de Type 9* | Primitive "confirmation" | → | Message de réponse |
| NOTE Pour les passerelles E/S, l'abstraction de l'interface de la pile communication de Type 9 est maintenue. | | | |

#### 7.1.5.5.7 Réception de messages

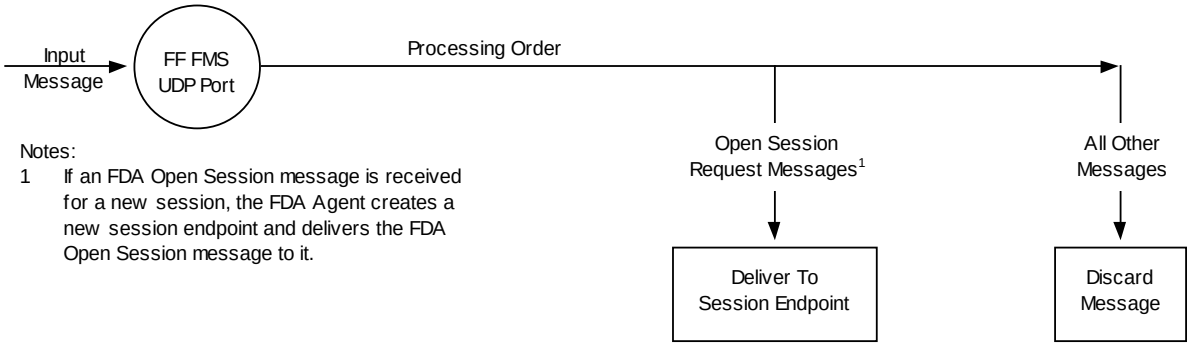
Après initialisation, l'agent FDA se met à l'écoute des ports SM et FDA ainsi que d'autres ports configurés pour des messages FDA. Il distingue et livre les types suivants de messages (voir Figure 11 à Figure 15).




**Légende**

| Anglais | Français |
|---|---|
| Input Message | Message d'entrée |
| FF SM UDP Port | Port UDP SM FF |
| Processing Order | Ordre de traitement |
| SM Message | Message SM |
| Deliver To SMK | Livrer au SMK |
| All Other Messages | Tous les autres messages |
| Discard Message | Rejeter le message |

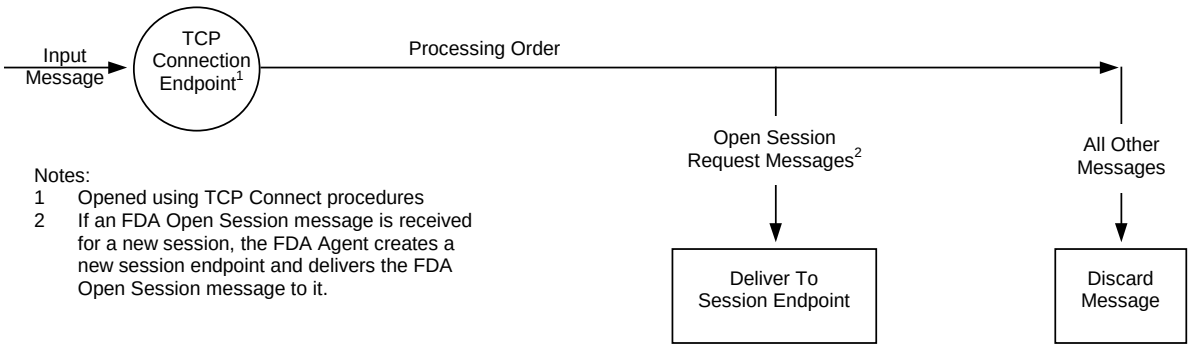
**Figure 11 – Ordre de traitement des messages aux ports SM FF**




**Légende**

| Anglais | Français |
|---|---|
| Input Message | Message d'entrée |
| FF SM UDP Port | Port UDP SM FF |
| Processing Order | Ordre de traitement |
| Notes:<br>If an FDA Open Session message is received for a new session, the FDA Agent creates a new session endpoint and delivers the FDA Open Session message to it. | Notes :<br>Si un message «Ouvrir une session FDA» est reçu pour une nouvelle session, l'Agent FDA crée une nouvelle pour le point d'extrémité et lui délivre le message « Ouvrir une session FDA » |
| Open Session Request Messages[1] | Messages de demande d'ouvrir la session |
| Deliver To Session Endpoint | Livrer à la session du point d'extrémité |
| All Other Messages | Tous les autres messages |
| Discard Message | Rejeter le message |

**Figure 12 – Ordre de traitement des messages aux ports FDA FF**

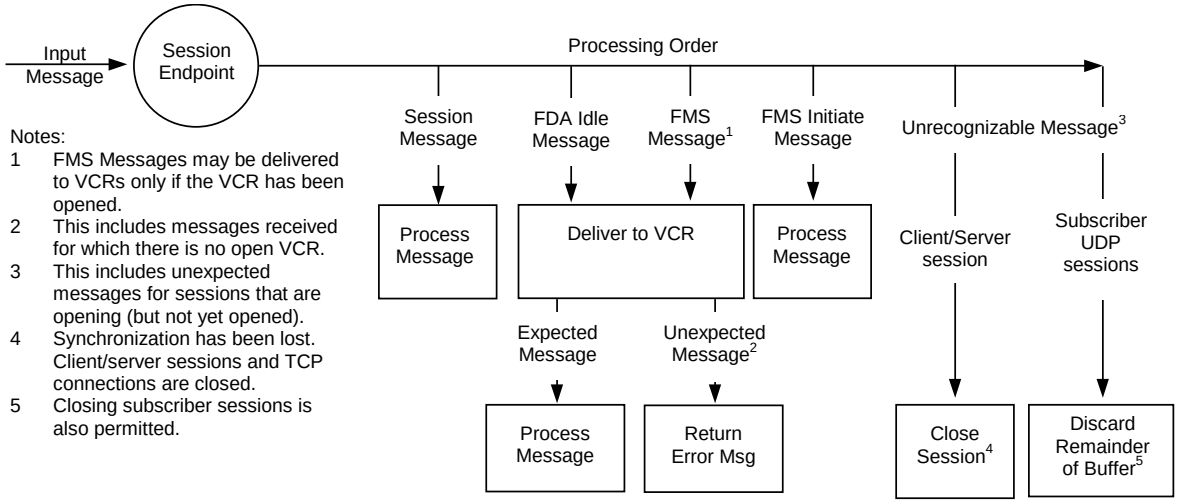



**Légende**

| Anglais | Français |
|---|---|
| Input Message | Message d'entrée |
| TCP Connection Endpoint[1] | Point d'extrémité de connexion TCP |
| Processing Order | Ordre de traitement |
| Open Session Request Messages[2] | Messages de demande d'ouvrir une session |
| All Other Messages | Tous les autres messages |
| Notes:<br>Opened using TCP Connect procedures.<br>If an FDA Open Session message is received for a new session, the FDA Agent creates a new session endpoint and delivers the FDA Open | Notes :<br>Ouvert en utilisant les procédures de connexion TCP<br>Si un message « ouvrir une session FDA » est reçu pour une nouvelle session, l'Agent FDA crée un nouveau point d'extrémité de session et lui |

| Anglais | Français |
|---|---|
| Session message to it | livre le message «Ouvrir la session FDA» |
| Deliver To Session Endpoint | Livrer à la session du point d'extrémité |
| Discard Message | Rejeter le message |

**Figure 13 – Ordre de traitement des messages aux connexions TCP FF**

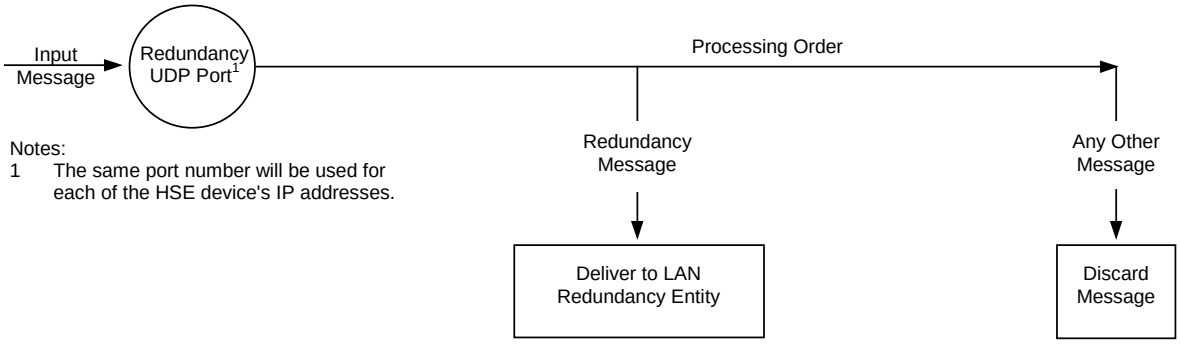



**Légende**

| Anglais | Français |
|---|---|
| Input Message | Message d'entrée |
| Session Endpoint | Point d'extrémité de session |
| Processing Order | Ordre de traitement |
| Notes:<br>FMS Messages may be delivered to VCRs only if the VCR has been opened.<br>This includes messages received for which there is no open VCR.<br>This includes unexpected messages for sessions that are opening (but not yet opened).<br>Synchronization has been lost. Client/server sessions and TCP connections are closed.<br>Closing subscriber sessions is also permitted. | Notes :<br>Les messages FMS ne peuvent être livrés aux VCR que si la VCR a été ouverte.<br>Cela inclut les messages reçus pour lesquels n'existe pas de VCR ouverte.<br>Cela inclut les messages inattendus pour les sessions qui s'ouvrent (mais ne sont pas encore ouvertes).<br>La synchronisation a été perdue. Les sessions client/server et les connexions TCP sont fermées<br>Il est également permis de fermer les sessions abonné |
| Session Message | Message de session |
| FDA Idle Message | Message « FDA Idle » |
| FMS Message[1] | Message[1] FMS |
| FMS Initiate Message | Message Initiate FMS |
| Unrecognizable Message[3] | Message[3] non-reconnaissable |
| Process Message | Traiter le message |
| Deliver to VCR | Livrer à VCR |
| Process Message | Traiter le message |
| Client/Server sessions | Sessions Client/Server |
| Subscriber UDP sessions | Sessions UDP abonné |
| Expected Message | Message attendu |

| Anglais | Français |
|---|---|
| Unexpected Message[2] | Message in attendu[2] |
| Process Message | Traiter le message |
| Return Error Msg | Retourner un message d'erreur |
| Close Session[4] | Fermer la session[4] |
| Discard Remainder of Buffer[5] | Rejeter le reste du buffer[5] |

**Figure 14 – Ordre de traitement des messages de session d'un point d'extrémité**

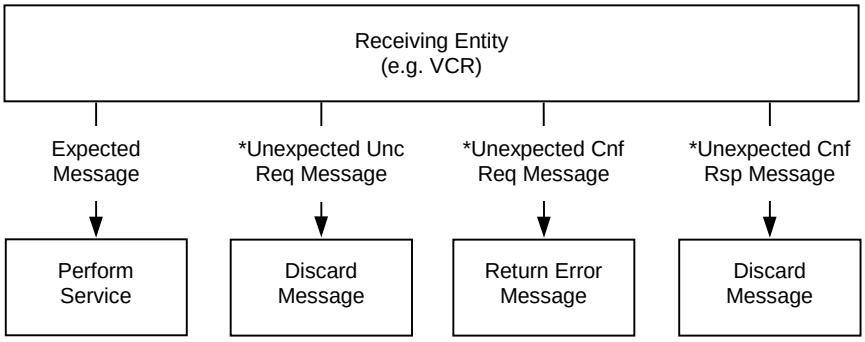



**Légende**

| Anglais | Français |
|---|---|
| Input Message | Message d'entrée |
| Redundancy UDP Port[1] | Port UDP de redondance[1] |
| Processing Order | Ordre de traitement |
| Notes:<br>The same port number will be used for each of the HSE device's IP addresses. | Notes :<br>Le même numéro de port sera utilisé pour chacune des adresses IP des équipements HSE. |
| Redundancy Message | Message de redondance |
| Any Other Message | Autre type de message |
| Deliver to LAN Redundancy Entity | Livrer à l'entité LAN Redundancy |
| Discard Message | Rejeter le message |

**Figure 15 – Ordre de traitement des messages FDA LAN redundancy port**

La Figure 16 récapitule le traitement des messages confirmés et non confirmés par l'entité réceptrice.




* An unexpected message is a message that is valid message that is not expected by the entity in its current state.

**Légende**

| Anglais | Français |
|---|---|
| Receiving Entity (e.g. VCR) | Entité réceptrice (par ex. VCR) |
| Expected Message | Message attendu |
| Unexpected Unc Req Message | *Message Unc Reg inattendu |
| Unexpected Cnf Req Message | *Message Cnf Reg inattendu |
| Unexpected Cnf Rsp Message | *Message Cnf Resp inattendu |
| Perform Service | Exécuter le service |
| Discard Message | Rejeter le message |
| Return Error Message | Renvoyer un message d'erreur |
| Discard Message | Rejeter le message |
| * An unexpected message is a message that is valid message that is not expected by the entity in its current state. | * Un message inattendu est un message valide qui n'est pas attendu par l'entité dans son état actuel. |

**Figure 16 – Traitement des messages par l'entité réceptrice**

#### 7.1.5.5.8 Démarrage et arrêt de temporisateurs

##### 7.1.5.5.8.1 Temporisateur VCR au repos

Le temporisateur VCR au repos (VCR Idle Timer) est utilisé pour maintenir les VCR client/serveur ouvertes en l'absence de demandes utilisateur. Le point d'extrémité VCR de type client redémarre le temporisateur chaque fois qu'il envoie un message. Les points d'extrémité de VCR de type serveur ne redémarrent pas ce temporisateur.

##### 7.1.5.5.8.2 Temporisateur de fermeture pour cause d'inactivité

Le temporisateur de fermeture pour cause d'inactivité (Inactivity Close Timer) commande la durée pendant laquelle les sessions des points d'extrémité VCR de type client et serveur restent ouvertes. Les sessions des points d'extrémité VCR de type client et serveur redémarrent ce temporisateur chaque fois qu'ils reçoivent un message.

##### 7.1.5.5.8.3 Temporisateur de retard d'émission

Le temporisateur de retard d'émission (Transmit Delay Timer) commande la durée pendant laquelle une session d'un point d'extrémité attend afin de transférer un message après l'avoir accepté pour transfert .Le temporisateur est redémarré lorsque le premier message est placé dans le buffer d'émission après que le buffer a été initialisé ou après qu'il a été vidé par l'émission précédente. Il est arrêté lorsque le buffer est rempli, amenant le buffer à être émis puis vidé.

#### 7.1.5.5.9 Bourrage de message

Les messages FDA sont conçus pour un traitement efficace en plaçant des champs de 16 bits et de 32 bits sur les limites appropriées de mots. Les champs de fin sont toutefois placés comme offset par rapport à la fin du message. Par conséquent, lorsque des données utilisateur sont présentes dans un message, l'alignement des limites des champs de fin est dépendant de la longueur des données utilisateur. En outre, lorsque des messages sont concaténés en un seul buffer pour le transfert, l'alignement des limites pour l'en-tête de message FDA est dépendant de la longueur du message qui précède.

Dans le but de maintenir des alignements efficaces des limites, il a été défini une fonctionnalité facultative de bourrage. Elle permet de configurer une session FDA en utilisant les bits Option du Message Header pour insérer des octets de bourrage après les données utilisateur et avant les champs de fin pour les services qui contiennent un index dans la demande ou dans la réponse correspondante. Pour insérer des octets, l'agent FDA expéditeur positionne les bits de bourrage Message Header Options pour indiquer le nombre des octets de bourrage qui sont insérés entre les données utilisateur et les champs de fin. La valeur de chaque octet de bourrage est mise à 0 binaire et le champ "Pad Length" de FDA Message Header Options est mis pour indiquer le nombre d'octets de bourrage.

Une erreur de protocole est d'insérer un bourrage qui n'aligne pas correctement la fin du message à la limite de 4 octets ou qui ne bourre pas la longueur de message jusqu'à un multiple de 4 octets. Cette restriction vise à prendre en charge les appareils qui ont des tailles de mots égales à 4 octets ou à 8 octets.

Une erreur de configuration est de relier un point d'extrémité de session qui envoie des messages bourrés à un autre qui est incapable de les recevoir.

NOTE 1 La NMIB indique si l'agent FDA prend en charge le bourrage comme expéditeur ou comme destinataire.

NOTE 2 Les messages LAN Redundancy et SM ne sont pas envoyés dans le cadre du contexte d'une session et ne sont donc pas bourrés.

### 7.1.5.6 Identification d'applications de Type 5 et de Type 9

#### 7.1.5.6.1 Généralités

Les messages FDA peuvent être envoyés ou reçus par les SMK/applications soit du Type 5, soit du Type 9. Pour identifier l'application/le SMK d'envoi ou de réception, il est utilisé une combinaison de l'adresse réseau et de l'adresse FDA. L'adresse FDA contient l'identificateur de liaison (Link Id) et le sélecteur (Selector). Chacun de ces identificateurs est débattu ci-dessous.

#### 7.1.5.6.2 Adresses réseau et adresses sous-réseau

Les adresses réseau sont utilisées comme paramètres adresse source et destination pour les appareils expéditeurs et destinataires. Les adresses source sont des adresses individuelles alors que les adresses destination peuvent être des adresses individuelles ou de multidiffusion. Lorsqu'elle est acheminée dans un message FDA, l'adresse réseau utilise un format d'adresse de 16 octets.

Les adresses sous-réseau sont définies comme étant les premiers N bits d'une adresse réseau individuelle d'un appareil, commençant par le bit de poids fort. Le nombre N est configurable et est maintenu dans la MIB.

Pour les messages de multidiffusion, l'appareil réel source du message n'est pas significatif, mais le sous-réseau qui est l'origine du message l'est. Par conséquent, le sous-réseau de l'expéditeur est utilisé pour qualifier l'adresse FDA contenue dans l'en-tête de message dans le cas des messages de multidiffusion.

Les adresses réseau définies dans la présente spécification sont énumérées dans le Tableau 89.

**Tableau 89 – Adresses réseau définies**

| Nom d'adresse | Usage de l'adresse | Valeur |
|---|---|---|
| Individuelle | Adresse d'une interface d'appareil | Attribuée par DHCP |
| Adresse réseau opérationnelle | Adresse d'une interface d'appareil utilisée dans les messages de service Find Tag Reply | Attribuée pendant l'affectation d'appareil |
| Multidiffusion de SM | Adresse de multidiffusion des SMK | Réservée |
| Adresse d'envoi en l'interface A de messages de diagnostic | Adresse de multidiffusion utilisée pour envoyer des messages de diagnostic LAN Redundancy à partir de l'interface réseau A. | Attribuée à l'aide du service LAN Redundancy Put Information |
| Adresse d'envoi en l'interface B de messages de diagnostic | Adresse de multidiffusion utilisée pour envoyer des messages de diagnostic LAN Redundancy à partir de l'interface réseau B. | Attribuée à l'aide du service LAN Redundancy Put Information |
| Adresse de multidiffusion d'éditeurs/source de rapports | Adresse de multidiffusion vers laquelle des éditeurs et des sources de rapports envoient leurs messages | Configurée dans la MIB |

Leur usage est montré dans le Tableau 90.

**Tableau 90 – Usage des adresses réseau**

| Usage de l'adresse | Type de l'adresse | Utilisée pour identifier |
|---|---|---|
| Source | Individuelle | le sous-réseau de l'expéditeur du message et l'appareil au sein du sous-réseau |
| Destination | Individuelle | le sous-réseau du destinataire prévu et l'appareil au sein du sous-réseau |
| Destination | Multidiffusion | Groupe de destinataires prévus |

#### 7.1.5.6.3 Le sélecteur (Selector)

##### 7.1.5.6.3.1 Généralités

Le sélecteur correspond au 16 bits de poids faible de l'adresse FDA. Il identifie un point d'extrémité de communication spécifique dans l'appareil identifié ou la liaison de Type 9 ou bien il est utilisé pour compléter une adresse fixe de 32 bits. Chaque type de session utilise un type différent de sélecteur. Le sélecteur peut être hiérarchique ou plat.

##### 7.1.5.6.3.2 Sélecteurs de serveur

###### 7.1.5.6.3.2.1 Généralités

Les sélecteurs de serveur (Server Selector) sont utilisés dans le champ adresse FDA de l'en-tête de message FDA dans les messages de demande et aussi dans les messages de réponse pour identifier un point d'extrémité CR utilisée pour accéder au VFD serveur d'une session client/serveur.

###### 7.1.5.6.3.2.2 Domaine d'application

Les sélecteurs de VCR de type serveur qui identifient une VCR de Type 9 sont qualifiés par l'identificateur de liaison (Link Id) et sont uniques dans la liaison de Type 9 identifiée. Les

sélecteurs de VCR de type serveur qui identifient une VCR de Type 5 ont un Link Id de 0 et sont uniques au sein de l'appareil. Ils ne sont pas tenus d'être uniques à travers les appareils.

#### 7.1.5.6.3.2.3 Utilisation

Les sélecteurs de point d'extrémité CR serveur sont utilisés uniquement pour des sessions Client/serveur (voir Tableau 91).

**Tableau 91 – Usage de sélecteurs de point d'extrémité dans les VCR de type serveur**

| Type de point d'extrémité de session | Usage |
|---|---|
| **Serveur** | Identifie le point d'extrémité VCR serveur au sein d'un appareil de Type 5 ou de Type 9. Dans le cas de sélecteurs de VCR de Type 5, deux valeurs sont utilisées. Une valeur est utilisée dans le message de demande de service Initiate pour identifier un point d'extrémité de VCR serveur générique pour le VFD associé. La seconde est le sélecteur pour le point d'extrémité de VCR qui avait été créé dynamiquement comme résultat du traitement de la demande de service Initiate. Ce sélecteur est l'indice d'OD de la VCR automatique dans la MIB. |

### 7.1.5.6.3.3 Sélecteurs éditeur/abonné

#### 7.1.5.6.3.3.1 Généralités

Les sélecteurs d'éditeur sont utilisés dans le champ adresse FDA de l'en-tête de message pour identifier le buffer de données de l'éditeur pour des sessions éditeur/abonné. Le paramètre indice contenu dans le message identifie le paramètre individuel édité.

#### 7.1.5.6.3.3.2 Domaine d'application

Les sélecteurs d'éditeur sont uniques dans la valeur de Link Id.

#### 7.1.5.6.3.3.3 Usage

Les sélecteurs d'éditeur sont utilisés uniquement pour des sessions Éditeur/Abonné (voir Tableau 92).

**Tableau 92 – Usage de sélecteurs de point d'extrémité dans les VCR de type éditeur**

| Type de point d'extrémité de session | Usage |
|---|---|
| **Éditeur** | Identifie le buffer de données au sein d'un appareil de Type 5 ou de Type 9. Ils peuvent être hiérarchiques, ou plats au sein de la valeur Link Id. Lorsqu'ils sont hiérarchiques, le premier octet identifie l'appareil de Type 9 alors que le second identifie le point d'extrémité de connexion DL au sein de l'appareil de Type 9. |
| NOTE Les points d'extrémité VCR de type éditeur et abonné sont stockés comme des VCR statiques dans la MIB. Leurs indices d'OD ne sont pas utilisés comme valeurs de sélecteur. | |

### 7.1.5.6.3.4 Sélecteurs de distribution de rapports

#### 7.1.5.6.3.4.1 Généralités

Les sélecteurs de source de rapports sont utilisés dans le champ adresse FDA de l'en-tête de message FDA pour identifier le VFD source de rapports.

#### 7.1.5.6.3.4.2 Domaine d'application

Les sélecteurs de source de rapports sont uniques dans la valeur de Link Id.

#### 7.1.5.6.3.4.3 Usage

Les sélecteurs de source de rapports sont utilisés uniquement pour des sessions de type Report Distribution (voir Tableau 93).

**Tableau 93 – Usage de sélecteurs de point d'extrémité dans les VCR de type source**

| Type de point d'extrémité de session | Usage |
|---|---|
| **Report Source** (source de rapports) | Identifie le VFD source de rapports au sein d'un appareil ou d'une liaison de Type 9. Ils peuvent être hiérarchiques, ou plats au sein de la valeur Link Id. Lorsqu'ils sont hiérarchiques, le premier octet identifie l'appareil de Type 9 alors que le second identifie le point d'accès aux services DL au sein de l'appareil de Type 9. |
| NOTE Les rapports et puits de rapports des points d'extrémité VCR de type source sont stockés comme des VCR statiques dans la MIB. Leurs indices d'OD ne sont pas restreints à des valeurs de 16 bits et ils ne sont pas non plus utilisés comme valeurs de sélecteur. | |

### 7.1.5.7 Utilisation de services sous-jacents sans connexion et orientés connexion

#### 7.1.5.7.1 Généralités

L'utilisation de services sans connexion est la valeur par défaut utilisée pour prendre en charge les sessions. Les services orientés connexion peuvent être utilisés comme une option pour des sessions client/serveur. La sélection du protocole devant être utilisé est effectuée par le client pu par l'application de configuration si le point d'extrémité client est configurée.

#### 7.1.5.7.2 Mode sans connexion

Les services sans connexion entre deux ou plusieurs points d'extrémité de session d'agent FDA sont orientés message. Par conséquent, chaque buffer de messages contient toujours un nombre entier de messages FDA. Les messages FDA ne sont jamais scindés à travers les buffers de messages.

Bien que les services sans connexion puissent être utilisés pour tous les types de session d'agent FDA, ils sont toujours utilisés pour prendre en charge les sessions d'agent FDA qui exigent une livraison en multidiffusion ou qui n'exigent pas de livraison de message séquentielle fiable et à flux commandé.

#### 7.1.5.7.3 Orienté connexion

Les services orientés connexion sont utilisés entre deux et seulement deux sessions de points d'extrémité Agent FDA. Les connexions sont seulement utilisées pour prendre en charge les sessions client/serveur agent FDA. Autrement dit, ils ne sont pas utilisés pour prendre en charge le transfert sans session de messages SM ou pour prendre en charge des transferts en multidiffusion.

Il convient que des services orientés connexion soient utilisés par les sessions agent FDA qui souhaitent échanger des données d'une façon fiable, séquentielle et à flux commandé.

#### 7.1.5.7.4 Messages concaténés

Les services sans connexion et orientés connexion transfère de façon transparente le contenu des buffers entre applications. Ne pas inspecter les contenus ni leur apporter des changements.

Les sessions FDA mettent leurs messages dans des buffers avant de les envoyer. La structure des messages FDA a été conçue pour permettre que des sessions mettent (concatènent) plus d'un message dans le même buffer pour le transfert. Cela peut améliorer l'utilisation du réseau et peut réduire le nombre d'interruptions au niveau des destinataires. Les messages SM FDA n'étant pas envoyés dans des sessions, ils ne sont donc pas concaténés dans des buffers.

Le nombre de messages qui peuvent être concaténés ensemble et être envoyés est limité par la taille du buffer d'envoi. Les buffers sont envoyés dès qu'ils sont remplis. Ils sont toujours envoyés en contenant un nombre entier de messages lorsque les services sans connexion sous-jacents sont utilisés. Lorsqu'il est au transfert utilisant des services sans connexion sous-jacents un message qui provoquerait un dépassement de capacité du buffer, le buffer est envoyé sans ce message, qui est inséré comme premier message dans le prochain buffer devant être envoyé. Lorsque des services orientés objet sous-jacents sont utilisés, les messages peuvent s'étaler sur les buffers, car leur transfert et leur livraison peuvent être orientés flux.

Afin d'éviter que les buffers n'attendent infiniment pour être remplis, les sessions des points d'extrémité contiennent un attribut appelé "Transmit Delay Time" (Durée de retard d'émission). Cet attribut dicte la durée pendant laquelle l'agent FDA attend que le buffer remplisse. S'il n'est pas rempli pendant cette durée, l'agent FDA transfère son contenu et prépare un nouveau buffer pour le prochain transfert.

#### 7.1.5.7.5 Numéros de port

##### 7.1.5.7.5.1 Généralités

Les ports identifient le point d'accès d'un ensemble à la couche sous-jacente. Il y a trois adresses de port utilisées par le FDA. L'une est utilisée exclusivement pour l'annonce d'appareil SM (SM Device Annunciation). Elle est appelée "Annunciation Port" (Port d'annonce"). Le deuxième port est utilisé pour l'accès SMK. Elle est appelée "SM Port" ("Port SM").

Le troisième port réservé est utilisé pour recevoir des messages de demande de service Open Session de FDA. Elle est appelée "FDA Port" ("Port FDA"). Les réponses positives à des messages de demande de service Open Session sont envoyées à partir de ports non utilisés précédemment et ces ports sont utilisés pour transférer tous les messages de session ultérieurs. Les réponses négatives à des messages de demande de service Open Session sont envoyées à partir du port FDA.

Les trajets de données entre applications sont identifiés par une combinaison adresse réseau/numéro de port source et destination.

##### 7.1.5.7.5.2 Domaine d'application

Les numéros de port sont uniques dans une adresse réseau individuelle. Le même numéro de port peut être utilisé simultanément, et sans confusion, pour des sockets sans connexion ou des sockets orientés connexion. Le protocole sous-jacent qualifie le numéro de port.

##### 7.1.5.7.5.3 Usage

Les types de session d'agent FDA utilisent des numéros de port conjointement aux adresses réseau (voir Tableau 94 à Tableau 103). Dans ces tableaux, les entrées dans les colonnes montrent les adresses auxquelles le point d'extrémité est attaché.

**Tableau 94 – Adresse réseau et numéros de port pour l'annonce d'appareil**

| Type de transfert | Le SMK d'appareil s'annonce à partir | → | L'AP de configuration est à l'écoute: |
|---|---|---|---|
| Multidiffusion | de l'adresse réseau d'appareil<br>de tout port non utilisé ou du port FDA réservé | | de l'adresse réseau en multidiffusion SM réservée<br>du port d'annonce réservé |

**Tableau 95 – Adresse réseau et numéros de port pour établir/éliminer les informations d'attribution et libérer l'adresse**

| Type de transfert | L'AP de configuration envoie et reçoit en utilisant | → ← | SMK est à l'écoute de et répond à partir de : |
|---|---|---|---|
| Monodiffusion en mode sans connexion | l'adresse réseau de l'AP de configuration<br>N'importe quel port non utilisé | | l'adresse réseau opérationnelle d'appareil<br>Port SM réservé |

**Tableau 96 – Adresse réseau et numéros de port pour le service SM Identify**

| Type de transfert | L'AP demandeur envoie la demande à partir de: | | Le SMK répondeur | |
|---|---|---|---|---|
| | | | est à l'écoute de | répond à partir de |
| Monodiffusion en mode sans connexion | l'adresse réseau du demandeur | → ← | l'adresse réseau opérationnelle d'appareil<br>Port SM réservé | l'adresse réseau opérationnelle d'appareil |
| Multidiffusion | N'importe quel port non utilisé | | de l'adresse réseau en multidiffusion SM réservée<br>Port SM réservé | Port SM réservé |

**Tableau 97 – Adresse réseau et numéros de port pour le service SM Find Tag**

| Type de transfert | L'AP demandeur envoie l'interrogation à partir de: | | Le SMK répondeur: | |
|---|---|---|---|---|
| | | | est à l'écoute de | répond à partir de |
| Monodiffusion en mode sans connexion | l'adresse réseau du demandeur | → ← | l'adresse réseau opérationnelle d'appareil<br>Port SM réservé | l'adresse réseau opérationnelle d'appareil |
| Multidiffusion | N'importe quel port non utilisé | | de l'adresse réseau en multidiffusion SM réservée<br>Port SM réservé | Port SM réservé |

**Tableau 98 – Adresse réseau et numéros de port pour clients et serveurs (partie 1)**

| Type de transfert | L'AP client ouvre la session à partir de: | | L'appareil serveur: | |
|---|---|---|---|---|
| | | | est à l'écoute de | répond à partir de |
| Monodiffusion en mode sans connexion | l'adresse réseau du client<br>N'importe quel port non utilisé | → ← | l'adresse réseau opérationnelle d'appareil<br>Port FDA réservé | l'adresse réseau opérationnelle d'appareil<br>N'importe quel port non utilisé (voir NOTE) |
| NOTE Également utilisé comme étant le port pour recevoir et envoyer tous les messages ultérieurs sur la session. Les réponses négatives sont retournées à partir du port FDA si le serveur est incapable d'acquérir un port pour répondre. | | | | |

**Tableau 99 – Adresse réseau et numéros de port pour clients et serveurs (partie 2)**

| Type de transfert | L'AP client ouvre la connexion TCP à partir de: | | L'appareil serveur: | |
|---|---|---|---|---|
| | | | est à l'écoute de | répond en utilisant |
| Orienté connexion | l'adresse réseau du client<br>N'importe quel port non utilisé | → ← | l'adresse réseau opérationnelle d'appareil<br>Port FDA réservé | l'adresse réseau opérationnelle d'appareil<br>la connexion TCP nouvellement établie (voir NOTE) |
| NOTE La connexion TCP est utilisée par les deux points d'extrémité pour envoyer et recevoir tous les messages sur la session, y compris les messages de demande et de réponse de service Open Session de FDA. | | | | |

**Tableau 100 – Adresse réseau et numéros de port pour éditeurs et abonnés**

| Type de transfert | L'appareil éditeur envoie à partir de: | → | L'appareil abonné est à l'écoute de: |
|---|---|---|---|
| Multidiffusion | l'adresse réseau opérationnelle d'appareil éditeur | | l'adresse réseau en multidiffusion configurée |
| | N'importe quel port non utilisé | | port configuré* |
| * Les adresses de multidiffusion et les numéros de port sont configurés comme étant l'adresse/port de destination pour les points d'extrémité de type éditeur. Les points d'extrémité de type abonné sont configurées pour recevoir des messages à ces adresses/ports. Le port configuré est attribué à partir de la plage non utilisée de numéros de port. Les numéros de port réservés ou les numéros de port servant à d'autres fins ne peuvent pas être utilisés. | | | |

**Tableau 101 – Adresse réseau et numéros de port pour le service Report Distribution**

| Type de transfert | L'appareil source de rapports envoie à partir de: | → | L'appareil puits de rapports est à l'écoute de: |
|---|---|---|---|
| Multidiffusion | l'adresse réseau opérationnelle de l'appareil source de rapports | | l'adresse réseau en multidiffusion configurée* |
| | N'importe quel port non utilisé | | port configuré* |
| * Les adresses de multidiffusion et les numéros de port sont configurés comme étant l'adresse/port de destination pour les points d'extrémité de type source de rapports. Les points d'extrémité de type puits de rapports sont configurées pour recevoir des messages à ces adresses/ports. Le port configuré est attribué à partir de la plage non utilisée de numéros de port. Les numéros de port réservés ou les numéros de port servant à d'autres fins ne peuvent pas être utilisés. | | | |

**Tableau 102 – Adresse réseau et numéros de port pour les informations relatives aux services LAN Redundancy Get and Put**

| Type de transfert | L'AP client ouvre la session à partir de: | | L'appareil serveur: | |
|---|---|---|---|---|
| | | | est à l'écoute de | répond à partir de |
| Unicast | l'adresse réseau opérationnelle du client | → ← | l'adresse réseau opérationnelle d'appareil | l'adresse réseau opérationnelle d'appareil |
| | N'importe quel port non utilisé | | le port LAN Redundancy attribué | le port LAN Redundancy attribué |

**Tableau 103 – Adresse réseau et numéros de port pour le diagnostic LAN redundancy**

| Type de transfert | L'appareil envoie à partir de: | → ← | Les appareils sont à l'écoute de: |
|---|---|---|---|
| Multidiffusion | l'adresse réseau opérationnelle d'appareil | | l'adresse de multidiffusion attribuée |
| | N'importe quel port non utilisé | | le port LAN Redundancy attribué |

### 7.1.6 Support pour la gestion de configuration

Les données de configuration pour l'agent FDA sont maintenues dans la MIB. L'agent FDA utilise également des structures de données de pont dans la MIB s'il se situe dans un appareil de liaison et n'est pas pris en charge par les fonctionnalités de pontage de Type 9. Il utilise les structures de données de pont pour accomplir des opérations de réédition et de transmission de rapports qui sont normalement accomplies par le pont

## 7.2 ASE

### 7.2.1 ASE Virtual Field Device (Appareil de terrain virtuel)

#### 7.2.1.1 Vue d'ensemble

Le VFD de Type 5 est le même que le VFD de Type 9, excepté pour l'objet VCR et le service Initiate. Ils remplacent ceux indiqués dans la spécification du Type 9. Tous les autres objets et services du Type 9 s'appliquent. Ils ne sont pas répétés ici.

#### 7.2.1.2 Spécifications de la classe-modèle des appareils de terrain virtuels simples

##### 7.2.1.2.1 Vue d'ensemble de la classe

Le présent paragraphe donne la définition de classe formelle pour la classe VCR. Les VCR sont analogues aux VCR de Type 9.

##### 7.2.1.2.2 Modèle formel de VCR

La classe VCR spécifie les attributs et les services de la VCR. Sa classe parente "top" indique le sommet de l'arbre de la classe FAL.

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE VFD** |
| **CLASS:** | | | **VCR** |
| **CLASS ID:** | | | non utilisé |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTS DE LA GESTION SYSTÈME:** | | | |
| 1 | (m) | Attribut: | VCR Id |
| 2 | (m) | Attribut: | Related AREP Id |
| 3 | (m) | Attribut: | VCR Type |
| 4 | (m) | Attribut: | VCR State |
| 5 | (m) | Attribut: | VCR User Id |
| 6 | (m) | Attribut: | FDA Address |
| 7 | (c) | Contrainte | VCR Type = LOCAL SUBSCRIBER<br>\|\| TYPE 9 PUBLISHER<br>\|\| LOCAL REPORT SINK<br>\|\| TYPE 9 REPORT SOURCE |
| 7.1 | (m) | Attribut: | Subnet Mask |
| 9 | (c) | Contrainte | VCR Type = LOCAL PUBLISHER |
| 9.1 | (m) | Attribut: | On Change Threshold |
| 7 | (c) | Contrainte | VCR Type = LOCAL CLIENT<br>\|\| TYPE 9 SERVER<br>\|\| LOCAL SERVER<br>\|\| TYPE 9 CLIENT<br>\|\| LD TYPE 9 MIB AGENT |
| 7.1 | (m) | Attribut: | Inactivity Close Time |
| **SERVICES:** | | | |
| 1 | (o) | Ops Service: | Initiate |

##### 7.2.1.2.3 Attributs de la gestion système

Les attributs suivants sont accessibles par le biais de System Management. Ils ne sont pas accessibles autrement.

**Local VCR Id**

Cet attribut identifie la VCR.

### Related AREP Id

Cet attribut est l'identificateur numérique des AREP connexes qui sont utilisés pour envoyer ou recevoir des messages. Cette valeur est 0 lorsque la VCR est utilisée pour rééditer ou transmettre des rapports entre deux AR de Type 9 dans un appareil de liaison. La valeur 0 signifie qu'il n'y a pas de point d'extrémité de session connexe.

### VCR Type

Cet attribut indique le type de la VCR. Chacun des types utilise l'attribut Related AREP (c'est-à-dire AREP connexe) pour acheminer des services. Sa valeur est telle que montrée dans le Tableau 104.

**Tableau 104 – Types de VCR**

| VCR Type (Type de VCR) | Utilisation |
|---|---|
| LOCAL CLIENT | La VCR reçoit des primitives "request" de services confirmés issues de l'AP. |
| LOCAL SERVER | La VCR livre des primitives "indication" de services confirmés à l'AP en vue de leur traitement. |
| LOCAL PUBLISHER | La VCR édite des données ayant leur origine dans l'AP. |
| LOCAL SUBSCRIBER | La VCR s'abonne à des données produites et les livre à l'AP. |
| LOCAL REPORT SOURCE | La VCR distribue des rapports ayant leur origine dans l'AP. |
| LOCAL REPORT SINK | La VCR reçoit des rapports et les livre à l'AP. |
| TYPE 9 SERVER[1] | La VCR est un serveur sur une liaison de Type 9 et un client sur un réseau de Type 5. La VCR reçoit des primitives "indication" de services confirmés issues de la liaison de Type 9 et les transmet sur le réseau de Type 5. |
| TYPE 9 CLIENT[1] | La VCR est un client sur une liaison de Type 9 et un serveur sur un réseau de Type 5. La VCR reçoit des primitives "indication" de services confirmés issues du réseau de Type 5 et les transmet sur une liaison de Type 9. |
| TYPE 9 SUBSCRIBER[1] | La VCR est un abonné sur une liaison de Type 9 et un éditeur sur un réseau de Type 5. La VCR reçoit des données produites issues de la liaison de Type 9 et les réédite sur le réseau de Type 5. |
| TYPE 9 PUBLISHER[1] | La VCR est un éditeur sur une liaison de Type 9 et un abonné sur un réseau de Type 5. La VCR reçoit des données produites issues du réseau de Type 5 et les réédite sur la liaison de Type 9. |
| TYPE 9 REPORT SINK[1] | La VCR est un puits de rapports sur une liaison de Type 9 et une source de rapports sur un réseau de Type 5. La VCR reçoit des rapports issus de la liaison de Type 9 et les transmet sur le réseau de Type 5. |
| TYPE 9 REPORT SOURCE[1] | La VCR est une source de rapports sur une liaison de Type 9 et un puits de rapports sur un réseau de Type 5. La VCR reçoit des rapports issus du réseau de Type 5 et les transmet sur la liaison de Type 9. |
| LD TYPE 9 MIB AGENT[1] | La VCR livre des primitives "indication" de services confirmés MIB de Type aux agents en vue de leur traitement (applicable seulement à des appareils de liaison). |

NOTE Les VCR de Type 9 ne sont présentes que dans des appareils de liaison.

### VCR State

Cet attribut spécifie l'état de la VCR. Les états valides sont définis dans la CEI 61158-6-5.

### VCR User Id

Cet attribut conditionnel est l'adresse FDA qui identifie l'utilisateur de la VCR. L'utilisation de cet attribut est montrée dans le Tableau 105.

**Tableau 105 – Utilisation de VCR user id**

| VCR Type (Type de VCR) | Contenu (représenté comme étant une adresse FDA): |
|---|---|
| LOCAL SERVER | L'adresse FDA contenue dans la demande Initiate de Type 9 utilisée pour ouvrir la VCR. |
| LD TYPE 9 MANAGEMENT AGENT | Liaison de Type 9 de l'agent de gestion. La partie N.S de l'adresse n'est pas utilisée et elle est mise à 0. |
| TYPE 9 SERVER | La VCR de type serveur de Type 9 utilisée pour recevoir des demandes issues de la liaison Type 9. |
| TYPE 9 CLIENT | La VCR de type client de Type 9 utilisée pour envoyer des demandes sur la liaison de Type 9. |
| TYPE 9 SERVER | La VCR de type serveur de Type 9 utilisée pour recevoir des demandes issues de la liaison Type 9. |
| TYPE 9 PUBLISHER | VCR de type éditeur de Type 9 utilisée pour éditer sur la liaison de Type 9. |
| TYPE 9 SUBSCRIBER | La VCR de type abonné de Type 9 utilisée pour s'abonner à des données sur la liaison de Type 9. |
| TYPE 9 REPORT SOURCE | VCR de type Report Source (source de rapports) du Type 9 utilisée pour envoyer des rapports sur la liaison de Type 9 |
| TYPE 9 REPORT SINK | VCR de type Report Sink (puits de rapports) du Type 9 utilisé pour recevoir des distributions de rapports issues de la liaison de Type 9. |

### FDA Address

Cet attribut spécifie une référence qui est acheminée avec chaque service à destination ou en provenance du/des point(s) d'extrémité de VCR correspondant(s). Ses valeurs sont dépendantes du VCR Type (voir Tableau 106).

**Tableau 106 – Utilisation de l'adresse FDA**

| VCR Type (Type de VCR) | Utilisation |
|---|---|
| LOCAL CLIENT | • L'identificateur de la VCR de type serveur associée ou de l'AREP de Type 9 associé. |
| LOCAL SERVER | • Le VCR Id du point d'extrémité LOCAL SERVER VCR. |
| LOCAL PUBLISHER | • Référence de "Published-Data" utilisée pour identifier le buffer de données produites.<br>• Cette adresse est prise dans l'espace adresse de DLCEP fixe de Type 9 de 32 bits.<br>• Le masque de sous-réseau de Type 5 de l'éditeur qualifie cette référence "Published-Data". |
| LOCAL SUBSCRIBER | • La référence du buffer de données de l'éditeur qui est "Subscribed-To" (c'est-à-dire auxquelles un abonnement est souscrit).<br>• Le masque de sous-réseau de Type 5 de l'éditeur qualifie cette référence "Subscribed-To". |
| LOCAL REPORT SOURCE | • La référence "Reporting VFD" utilisée pour identifier le VFD envoyant les messages de rapports.<br>• Cette adresse est prise dans l'espace adresse de DLSAP fixe de Type 9 de 32 bits.<br>• Le masque de sous-réseau de Type 5 qualifie cette référence "Reporting VFD".<br>• Cet identificateur de VFD qualifie la valeur du paramètre Index (indice) contenue dans les messages de rapports dans les paramètres spécifiques à un service (chaque VFD source peut envoyer un ou plusieurs messages de rapports, chacun ayant un Index différent. |
| LOCAL REPORT SINK | Non utilisé. Une seule et même VCR LOCAL REPORT SINK peut recevoir des rapports issus de plus d'une source de rapports. Par conséquent, les APDU reçues en provenance de l'AREP connexe peuvent contenir une valeur de FDA Address différente. |
| TYPE 9 SERVER | • L'identificateur numérique de l'AREP serveur de Type 9. |
| TYPE 9 CLIENT | Non utilisé. |
| TYPE 9 PUBLISHER | • Si l'identificateur d'AREP connexe (Related AREP Id) est non nul, il s'agit du *FDA Address* dans les APDU de l'éditeur de Type 5 qui est "Subscribed-To". Le masque de sous-réseau de Type 5 de l'éditeur qualifie cette référence "Subscribed-To".<br>• Si le Related AREP Id est nul, cette valeur est le DLCEP de Type 9 utilisé par la VCR de type abonné de Type 9 pour des données produites.<br>• Les données reçues sont reproduites sur la liaison de Type 9 en utilisant l'attribut VCR User Id. |
| TYPE 9 SUBSCRIBER | • L'adresse de liaison de données de Type 9 du buffer de données produites. |
| TYPE 9 REPORT SOURCE | • Non utilisé. Une seule et même VCR TYPE 9 REPORT SOURCE peut recevoir des rapports issus de plus d'une source de rapports. Par conséquent, les APDU reçues en provenance de l'AREP connexe peuvent contenir une valeur de FDA Address différente. |
| TYPE 9 REPORT SINK | • Non utilisé. Une seule et même VCR TYPE 9 REPORT SINK peut recevoir des rapports issus de plus d'une source de rapports. Par conséquent, les rapports reçus en provenance de la VCR de Type 9 peuvent contenir une adresse de source de Type 9 différente. |

### Type 5 Subnet Mask

Cet attribut identifie le sous-réseau d'adresse FDA et sert à le qualifier pour des primitives "indication" de services sur la VCR.

### On Change Threshold

Cet attribut conditionnel n'est présent que dans les VCR "LOCAL PUBLISHER". Il spécifie le pourcentage de variation devant être excédé avant que la valeur des données produites ne soit considérée comme ayant changé. Si sa valeur est 0, chaque valeur est produite en utilisant le service Information Report de Type 9, indépendamment du fait qu'elle ait changé ou non. La valeur par défaut est 0.

Si sa valeur n'est pas 0, les valeurs modifiées sont produites en utilisant le service Information Report On Change du Type 5. La fonctionnalité On Change opère comme suit. La différence acceptable est calculée en multipliant la plage de données produites, spécifiée par l'application, par le pourcentage On Change Threshold. Si l'étendue de données entre les valeurs supérieure et inférieure était de 20 et le pourcentage On Change Threshold de 10%, la différence acceptable serait de 2. Lorsqu'une valeur de variable produite est disponible, la

valeur absolue de sa différence avec la dernière valeur produite est calculée. Si la nouvelle valeur est 17 et la précédente 14, la différence est 3. Si cette différence est supérieure à la différence acceptable (3 est plus grand que 2), la valeur est considérée comme ayant changé et est présentée en vue du transfert.

Si la valeur n'a pas été transférée pendant le nombre "refresh rate" de cycles d'exécution de l'éditeur, la valeur la plus récente est transférée. L'attribut "refresh rate" (taux de rafraîchissement) est contenu dans la MIB. La prise en charge de la fonctionnalité On Change est facultative, bien que la prise en charge de cet attribut soit obligatoire.

NOTE Il n'est pas spécifié si, oui ou non, cette fonctionnalité est mise en œuvre. Si elle est mise en œuvre dans l'AE, il faut que l'AE soit capable de décoder la partie "données" du message pour déterminer la valeur qui est produite. Si l'application met en œuvre cette fonctionnalité, il faut qu'elle lise cette valeur de l'attribut pour déterminer de présenter, oui ou non; les données pour l'édition.

**Inactivity Close Time**

Cet attribut commande la durée pendant laquelle le point d'extrémité de VCR de type client reste ouvert sans recevoir de message et il spécifie aussi la durée que le point d'extrémité de VCR de type client utilise pour commander l'envoi de messages de repos. Sa valeur est héritée de son AREP connexe.

#### 7.2.1.2.4 Services

**Initiate**

Ce service facultatif est utilisé pour ouvrir la VCR pour une relation entre applications.

### 7.2.1.3 Spécification du service ASE Virtual Field Device (appareil de terrain virtuel)

#### 7.2.1.3.1 Services pris en charge

#### 7.2.1.3.2 Ce paragraphe contient la définition de services qui sont propres à cet ASE Service Initiate

##### 7.2.1.3.2.1 Vue d'ensemble du service

Ce service est utilisé pour ouvrir le point d'extrémité VCR distante. Le point d'extrémité distante peut se situer dans un appareil de Type 5 ou dans un appareil de Type 9 qui est connecté à l'appareil de Type 5 par un appareil de liaison.

##### 7.2.1.3.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 107.

**Tableau 107 – Initiate**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| FDA Address | M | M | | |
| Connect Option | U | | | |
| Access Protection Supported Calling | M | M (=) | | |
| Password Calling | M | M (=) | | |
| Access Groups Calling | M | M (=) | | |
| Version OD Calling | M | M (=) | | |
| Profile Number Calling | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| | | | | |
| Result(+) | | | S | S (=) |
| VCR Id | | | C | C (=) |
| Version OD Called | | | M | M (=) |
| Profile Number Called | | | M | M (=) |
| | | | | |
| Result(–) | | | S | S (=) |
| VCR Id | | | C | C (=) |
| Error Class and Code | | | M | M (=) |
| Additional Code | | | U | U (=) |
| Additional Description | | | U | U (=) |

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.

### Argument

Ce paramètre contient les paramètres de l'invocation du service.

### FDA Address

Ce paramètre identifie le point d'extrémité VCR distante. Sa valeur est dépendante de la valeur de Connect Option

### Connect Option

Ce paramètre indique l'option qui est à utiliser pour ouvrir la VCR. Ses valeurs sont énumérées dans le Tableau 108.

**Tableau 108 – Connect option**

| Connect Option | Value | Description |
|---|---|---|
| VCR Selector | 1 | Cette option permet à l'application demandeuse de fournir le VCR Selector (sélecteur de VCR) qui est à utiliser pour se connecter au VFD. Si le VFD est un VFD de Type 5, le VCR Selector est un sélecteur générique pour accéder au VFD. Si le VFD est un VFD de Type 9, le VCR Selector est l'identificateur de l'AREP client de Type 9 qui est à utiliser pour accéder à l'appareil de Type 9. Cet AREP client de Type 9 ne peut pas être associé au point d'extrémité VCR normalisé d'accès à la MIB dans l'appareil de Type 9. |
| MIB [1] | 2 | Cette option permet à l'application demandeuse d'indiquer qu'elle souhaite se connecter à un VFD de MIB de Type 9 ou de Type 5. La partie "link id" du champ FDA Address indique si, oui ou non, la MIB est dans l'appareil de Type 9 ou de Type 5. |
| FBAP [2] | 3 | Cette option permet à l'application demandeuse d'indiquer qu'elle souhaite se connecter à un VFD Function Block Application Process (Bloc fonctionnel processus d'application sde (FBAP)) de Type 9 ou de Type 5. La partie "link id" du champ FDA Address indique si, oui ou non, la MIB est dans l'appareil de Type 9 ou de Type 5. La partie "node id" du champ FDA Address identifie le nœud contenant le FBAP, alors que la partie "selector" du champ FDA Address est un nombre ordinal qui commence à “1” pour indiquer celui des FBAP auquel il faut que l'accès ait lieu. |
| NOTE 1 La MIB n'est accessible qu'en utilisant ce service Connect.<br>NOTE 2 Le FBAP peut être accessible également en utilisant l'option VCR Selector Connect. | | |

### Access Protection Supported Calling

Ce paramètre spécifie la valeur de l'attribut Access Protection Supported du VFD appelant s'il en a un. S'il n'en a pas, sa valeur est null (vide).

### Password Calling

Ce paramètre spécifie le mot de passe devant être utilisé pour l'accès à tous les objets du serveur sur cette VCR. Si l'accès avec mot de passe n’est pas à utiliser sur cette VCR, sa valeur est null

### Access Groups Calling

Ce paramètre spécifie l'appartenance du client comme membre dans des groupes d'accès spécifiques. L'appartenance comme membre s'applique pour l'accès à tous les objets du serveur de cette VCR.

### Version OD Calling

Ce paramètre spécifie la version d'OD du client. Sa valeur est null (vide) si le client ne contient pas d'OD.

### Profile Number Calling

Ce paramètre spécifie la valeur de l'attribut Profile Number du VFD appelant s'il en a un. S'il n'en a pas, sa valeur est null (vide).

### Physical Device Tag

Ce paramètre spécifie la valeur de Physical Device Tag (indicateur d’appareil physique) du SMK de l'appareil qui contient le VFD appelé s'il y en a un. S'il n'y en a pas, sa valeur est null (vide).

### Result (+)

Ce paramètre de type sélection indique que la demande de service a réussi.

### VCR Id

Cet attribut conditionnel identifie le point d'extrémité VCR à partir de laquelle la réponse a été envoyée.

### Version OD Called

Il convient que ce paramètre spécifie la version d'OD du serveur.

**Profile Number Called**

Ce paramètre spécifie la valeur de l'attribut Profile Number du VFD appelé.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**VCR Id**

Cet attribut conditionnel identifie le point d'extrémité VCR à partir de laquelle la réponse a été envoyée. Il est présent lorsque l'AREP sous-jacent prend en charge plusieurs VCR ouvertes simultanément.

**Error Class and Code**

Il convient que ce paramètre fournisse la cause de l'échec.

**Additional Code**

Ce paramètre facultatif fournit un code complémentaire qui est associé à l'erreur.

**Additional Description**

Ce paramètre facultatif fournit 16 octets d'informations complémentaires qui sont associées à l'Erreur.

#### 7.2.1.3.2.3 Procédure du service

Ce service fonctionne par le biais d'une file d'attente. La procédure de service confirmé spécifiée en 4.6 s'applique à ce service. Le traitement spécifique au service est comme suit:

Si l'agent FDA destinataire est incapable de localiser la VCR identifiée par l'adresse FDA ou si la VCR n'est pas un serveur AP ou un client de Type 9 ou bien si le Physical Device Tag dans la demande ne concorde pas avec le Physical Device Tag du SMK de l'appareil, l'agent FDA retourne une réponse négative contenant l'adresse FDA issue de la demande.

Si la demande identifie un VFD local, l'agent FDA construit un nouveau point d'extrémité VCR LOCAL SERVER et lui livre la demande pour traitement.

Si la demande identifie un TYPE 9 CLIENT, l'agent FDA construit un nouveau TYPE 9 CLIENT local et l'associe à l'AREP client de Type 9. Si l'AREP client de Type 9 n'est pas déjà ouvert, l'agent FDA l'ouvre en utilisant le service d'ASE d'AP Iniitiate de Type 9. Cela permet à plus d'un TYPE 9 CLIENT d'être associé à l'AREP client de Type 9.

Si la demande est acceptée, la réponse retournée au client par l'agent FDA contient le VCR Id de la VCR locale nouvellement créée. Cet Id est à utiliser par le client pour les accès ultérieurs sur la VCR.

Si la demande est rejetée, l'agent FDA supprime l'objet VCR s'il en a créé un et retourne une réponse négative.

Si la VCR a été ouverte à une MIB sur une AR qui ne prend pas en charge les demandes de mise à jour, l'agent FDA serveur émet une réponse négative s'il reçoit une demande de mise à jour provenant du client sans livrer l'indication de service à la MIB de gestion système.

### 7.2.2 ASE System management kernel

#### 7.2.2.1 Vue d'ensemble

Le modèle de noyau de gestion système de la FAL définit un objet qui prend en charge un agent de gestion système dans un appareil. L'entité de gestion système est appelée «System Management Kernel (Noyau de gestion de système (SMK)) parce qu'elle fournit des

fonctionnalités de base pour ajouter l'appareil au réseau ou l'en supprimer et pour localiser les objets au sein de l'appareil.

#### 7.2.2.2 Spécification de la classe System Management Kernel de la FAL

##### 7.2.2.2.1 Modèle formel de System Management Kernel (noyau de gestion système)

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE SYSTEM MANAGEMENT KERNEL** |
| **CLASS:** | | | **SYSTEM MANAGEMENT KERNEL** |
| **CLASS ID:** | | | non utilisé |
| **PARENT CLASS:** | | | TOP |
| **ATTRIBUTES:** | | | |
| 1 | (m) | Attribut-clé: | Device Id |
| 2 | (m) | Attribut: | Physical Device Tag |
| 3 | (m) | Attribut: | SMK State |
| 4 | (m) | Attribut: | Duplicate Detection State |
| 5 | (m) | Attribut: | LAN Redundancy Socket Address |
| 6 | (m) | Attribut: | Operational Network Address |
| 7 | (m) | Attribut: | Annunciation Repeat Time |
| 8 | (m) | Attribut: | Device Type |
| 9 | (m) | Attribut: | Device Redundancy State |
| 10 | (m) | Attribut: | Device Index |
| 11 | (m) | Attribut: | Max Device Index |
| 12 | (m) | Attribut: | SMK Attributes Version Number |
| 13 | (m) | Attribut: | Device Version Number |
| 14 | (m) | Attribut: | Operational Powerup |
| **SERVICES:** | | | |
| 1 | (m) | OpsService: | Find Tag Query |
| 2 | (m) | OpsService: | Find Tag Reply |
| 3 | (m) | OpsService: | Identify |
| 4 | (m) | OpsService: | Device Annunciation |
| 5 | (m) | OpsService: | Set Assignment Info |
| 6 | (m) | OpsService: | Clear Assignment Info |
| 7 | (m) | OpsService: | Clear Address |

##### 7.2.2.2.2 Attributs

**Device Id**

Cet attribut-clé spécifie l'identification spécifique à un fournisseur pour l'appareil.

**Physical Device Tag**

Cet attribut spécifie le nom (étiquette) administré de site attribué à l'appareil.

**SMK State**

L'attribut spécifie l'état opérationnel du SMK. Ses valeurs sont:

– No Tag

**Duplicate Detection State**

L'attribut spécifie si, oui ou non, le SMK a détecté un autre appareil avec son Physical Device Tag (PD Tag) ou son Device Index. Ses valeurs sont:

– No Duplicates Detected (Pas de doublons détectés)

– Duplicate Pd Tag Detected (Doublon de PD Tag détecté)

– Duplicate Device Index Detected (Doublon d'Index Device détecté)
– Duplicate Pd Tag and Device Index Detected (Doublons de PD Tag et d'Index Device détectés)

**LAN Redundancy Socket Address**

L'attribut spécifie l'adresse socket utilisée par l'appareil pour recevoir l'APDU LAN Redundancy.

**Operational Network Address**

Cet attribut spécifie l'adresse réseau utilisée par l'appareil. Plus d'une adresse peut être utilisée lorsque la redondance de réseau local (LAN) est prise en charge par l'appareil. Lorsque plus d'une adresse est prise en charge, une seule est désignée comme étant l'Operational Network Address (adresse réseau opérationnelle).

**Annunciation Repeat Time**

Cet attribut spécifie les millisecondes entre les messages d'annonce. Sa valeur par défaut est 15 000 (15 s).

**Device Type**

Cet attribut spécifie le type d'appareil et sa capacité de redondance. La liste des types valides inclut Type 9 Linking Device (appareil de liaison de Type 9), Type 9 Device (appareil de Type 9), Type 5 Device (appareil de type 5), Gateway Device (appareil passerelle).

La capacité de redondance d'un appareil spécifie la capacité de l'appareil de fonctionner comme un équipement d'un ensemble d'appareils redondants. Il faut que tous les appareils dans l'ensemble aient la même valeur de cet attribut configuré. Trois valeurs sont possibles, à savoir None (aucune), External Control (commande externe) et Automatic (automatique).

External Control indique que les appareils de cet ensemble sont capables de fonctionner soit comme le primaire, soit comme le secondaire (de secours). Un agent externe est requis pour synchroniser les données d'appareil, détecter une défaillance du primaire et, après détection, émettre une demande de service Set Assignment Info vers l'un des secondaires pour changer son état Device Redundancy State de secondaire à primaire.

Automatic indique que l'appareil est capable de se synchroniser avec ses partenaires redondants, de détecter une panne dans le primaire et de sélectionner automatiquement un nouveau primaire pour survivre à la panne.

**Device Redundancy State**

Cet attribut spécifie l'état réel de redondance de l'appareil. Si l'appareil ne participe pas à la redondance, sa valeur est None. S'il y participe comme appareil commandé externe, sa valeur est soit "external control – primary", soit "external control – secondary". S'il y participe comme appareil automatique, ses valeurs sont "automatic – primary" ou "automatic – secondary". Cette dernière valeur indique qu'il faut que les appareils de cet ensemble sélectionnent d'eux-mêmes lequel est le primaire. Le service Set Assignment Info peut être utilisé pour attribuer la valeur de cet attribut. Il convient que la valeur attribuée soit compatible avec l'attribut Device Type.

**Device Index**

Cet attribut est un numéro administré de site qui identifie l'appareil. Cet indice est unique au sein d'un sous-réseau FAL.

**Max Device Index**

Cet attribut définit la Valeur maximale que l'indice d'appareil peut avoir. Il indique effectivement le nombre maximal d'appareils qui peuvent être configurés dans un sous-réseau FAL. Ce nombre est un multiple de 32.

**SMK Attributes Version Number**

Cet attribut est le numéro de version pour les valeurs d'attribut qui précèdent cet attribut. Chaque fois que les valeurs de l'attribut qui précède changent, le numéro de version est incrémenté de 1. Ce numéro de version ne change que si la valeur de l'un des attributs couverts change. Si l'appareil s'initialise à la mise sous tension, la valeur du numéro de version est mise à zéro. Si l'appareil prend en charge Operational Powerup et celui-ci est mis à "true", le numéro de version d'annonce peut rester le même si les attributs qu'il couvre passent par le cycle de puissance sans changer. La valeur de SMK State peut changer si l'appareil prend en charge la synchronisation du temps et le statut de synchronisation du temps change, ce qui fait changer la valeur du numéro de version annoncée.

**Device Version Number**

Cet attribut est le numéro de version qui représente l'état composé des informations statiques dans les applications contenues dans l'appareil. Chaque temps qu'une modification est apportée aux données statiques de n'importe quelle application dans l'appareil, ce numéro de version est incrémenté de 1. Sa valeur s'initialise à zéro si l'appareil est initialisé. La manière dont le SMK détermine qu'il s'est produit un changement relève d'une initiative locale.

**Operational Powerup**

Cet attribut Boolean indique, lorsqu'il est true, que les données de SMK sont à stocker dans une mémoire non volatile afin qu'elles soient conservées à travers un cycle de puissance.

#### 7.2.2.2.3 Services

Tous les services définis pour cette classe sont obligatoires.

**Find Tag Query**

Ce service non confirmé sert à rechercher un objet dans le réseau.

**Find Tag Reply**

Ce service non confirmé est utilisé comme la réponse au service EMFind Tag Query. Il retourne une liste d'identificateurs de VCR qui peuvent être utilisés dans le service Initiate pour accéder à l'objet interrogé.

**Identify**

Ce service confirmé est utilisé pour demander au SMK de s'identifier lui-même.

**Device Annunciation**

Ce service non confirme est utilisé pour annoncer périodiquement la présence du SMK sur le réseau.

**Clear Address**

Ce service non confirmé est utilisé pour demander à un SMK de dégager son adresse réseau.

**Set Assignment Info**

Ce service confirmé est utilisé pour positionner un sous-ensemble sélectionné de valeurs d'attributs de SMK.

**Clear Assignment Info**

Ce service confirmé est utilisé pour effacer un sous-ensemble sélectionné de valeurs d'attributs de SMK.

### 7.2.2.3 Spécification des services de l'ASE System Management Kernel

#### 7.2.2.3.1 Services pris en charge

Les services de SMK opèrent directement sur les services de sockets. Ils n'utilisent pas d'AR.

#### 7.2.2.3.2 Service Find tag query

##### 7.2.2.3.2.1 Vue d'ensemble du service

Ce service non confirmé est utilisé pour envoyer une interrogation à un ou SMK pour localiser un objet sur le réseau.

##### 7.2.2.3.2.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 109.

**Tableau 109 – Paramètres du service Find tag query**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| Invoke ID | U | U (=) |
| Source Address | | M |
| Destination Address | M | M (=) |
| SMK ID | M | M (=) |
| Query Type | M | M (=) |
| Physical Device Tag | C | C (=) |
| VFD Tag | C | C (=) |
| Application Object Tag | C | C (=) |
| Element ID | C | C (=) |
| VFD Reference | C | C (=) |
| Device Index | C | C (=) |

**Argument**

L'argument contient les paramètres de la demande de service.

**Source Address**

Ce paramètre est l'adresse IP à partir de laquelle la demande de service a été envoyée. Il est utilisé par le répondeur pour retourner la réponse.

**Destination Address**

Ce paramètre est l'adresse à laquelle la demande de service est à envoyer. Cette adresse peut être une Individual Device Address (adresse d'appareil individuelle) ou la SMK Multicast Address (adresse en multidiffusion de SMK).

**SMK ID**

Ce paramètre identifie les SMK destinés à répondre à l'interrogation. Les valeurs possibles dépendent de la valeur de Destination Address. Elles sont montrées dans le Tableau 110.

**Tableau 110 – SMK ID**

| Valeur de Destination Address | Valeur de SMK ID |
|---|---|
| Individual Device Address (Adresse d'appareil individuelle) | SMK d'appareil |
| Individual Device Address | SMK d'appareil et tous les SMK de Type "X" attachés à l'appareil adressé (n'inclut pas les SMK de Type “X” SMK contenus dans l'appareil) |
| Individual Device Address | SMK d'appareil et tous les SMK de Type "X" sur une liaison spécifique de Type X attachés à l'appareil adressé (n'inclut pas les SMK de Type “X” SMK contenus dans l'appareil) |
| Individual Device Address | SMK spécifique de Type "X" accessible par le biais de l'appareil adressé |
| SMK Multicast Address (Adresse en multidiffusion de SMK) | Tous les SMK |
| SMK Multicast Address | Tous les SMK et tous les SMK de Type "X" (n'inclut pas les SMK de Type “X” contenus dans l'appareil) |
| SM Multicast Address | SMK d'appareil et tous les SMK de Type "X" sur une liaison spécifique de Type X attachés à l'appareil adressé (n'inclut pas les SMK de Type “X” SMK contenus dans l'appareil) |

**Query Type**

Ce paramètre sélectionne le type d'interrogation. Les types suivants s'excluent mutuellement:

– PD Tag query, Primary device
– PD Tag query, Secondary device
– VFD Tag query (exige également PD Tag)
– Application Object Tag query
– Element ID query (exige également l'Application Object Tag)
– VFD Reference query (exige également PD Tag)
– Device Index query

**Physical Device Tag (Étiquette d'appareil physique)**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient le Physical Device Tag, indiquant qu'il faut que l'interrogation localise un appareil physique. Il est présent lorsque le Query Type indique Physical Device Tag query ou VFD Tag query.

**VFD Tag**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient le VFD Tag, indiquant qu'il faut que l'interrogation localise un VFD. Il est présent lorsque le Query Type indique VFD Tag query.

**Application Object Tag**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient l'indicateur d'un objet application, indiquant qu'il faut que l'interrogation localise un objet marqué. Il est présent lorsque le Query Type indique Application Object Tag query ou Element ID query.

**Element ID**

Ce paramètre conditionnel, lorsqu'il est présent, est utilisé pour localiser un élément d'un Application Object Tag. Lorsqu'il est présent, l'Application Object Tag est également présent. Il contient une référence à un élément de l'objet application marqué. Il est présent lorsque le Query Type indique Element ID query.

**VFD Reference**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient une référence numérique à un VFD, indiquant qu'il faut que l'interrogation localise un VFD. Il est présent lorsque le Query Type indique VFD Reference query ou VFD Tag query.

**Device Index**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient le Device Index, indiquant qu'il faut que l'interrogation localise un appareil par son Device Index. Il est présent lorsque le Query Type indique Device Index query.

#### 7.2.2.3.2.3 Procédure du service

La procédure de service non confirmé spécifiée en 4.6 s'applique à ce service.

### 7.2.2.3.3 Service Find tag reply

#### 7.2.2.3.3.1 Vue d'ensemble du service

Ce service non confirmé est utilisé pour envoyer une réponse à l'émetteur d'un service Find Tag Query. Il contient des informations qui peuvent être utilisées pour établir une AR et une VCR pour accéder à l'objet interrogé.

#### 7.2.2.3.3.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 111.

**Tableau 111 – Paramètres du service Find tag reply**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| Invoke ID | U | U (=) |
| Source Address | | M |
| Destination Address | M | M (=) |
| Duplicate Detection State | M | M (=) |
| Query Type | M | M (=) |
| Queried Object Network Address | M | M (=) |
| Queried Object Link Address | M | M (=) |
| Queried Object Type "X" Device Address | C | C (=) |
| Queried Object Device ID | M | M (=) |
| Queried Object PD Tag | M | M (=) |
| Queried Object VFD Reference | C | C (=) |
| Queried Object OD Version Number | C | C (=) |
| Queried Object Numeric Id | C | C (=) |
| Duplicate Detection State | M | M (=) |
| List of VCR References | C | C (=) |

**Argument**

L'argument contient les paramètres de la demande de service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle ce service a été envoyé.

**Destination Address**

Ce paramètre est l'adresse IP à partir de laquelle le service Find Tag Query concordant a été envoyé.

**Duplicate Detection State**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom. Il renvoie au SMK de l'appareil répondeur.

**Query Type**

Ce paramètre sélectionne le type d'interrogation. Les types suivants s'excluent mutuellement:

– PD Tag query, Primary device
– PD Tag query, Secondary device
– VFD Tag query (exige également PD Tag)
– Function Block Tag query
– Element ID query (exige également FB Tag)
– VFD Reference query (exige également PD Tag)
– Device Index query

**Queried Object Network Address**

Ce paramètre spécifie l'adresse réseau utilisée pour accéder à l'objet interrogé.

**Queried Object Address**

Ce paramètre est la partie Link ID de l'adresse devant être utilisée pour accéder à l'objet interrogé. Si l'objet interrogé est contenu dans un appareil, sa valeur est 0. Si l'objet interrogé est situé dans un appareil de Type "X" sur une liaison de Type "X" qui est raccordé à l'appareil répondeur, cette valeur est l'adresse de liaison de Type “X”.

**Queried Object Type "X" Device Address**

Ce paramètre conditionnel est présent si l'objet interrogé est situé dans un appareil de Type "X" sur une liaison de Type "X" qui est raccordé à l'appareil répondeur. Il contient l'adresse de l'appareil de Type "X" utilisée pour accéder à l'appareil de Type "X".

**Queried Object Device ID**

Ce paramètre spécifie la valeur de l'attribut Device ID de SMK pour l'appareil qui contient l'objet interrogé.

**Queried Object PD Tag**

Ce paramètre spécifie la valeur de l'attribut PD Tag de SMK pour l'appareil qui contient l'objet interrogé.

**Queried Object VFD Reference**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient une référence numérique du VFD de l'objet interrogé. Il est présent pour toutes les interrogations, à l'exception des interrogations Physical Device Tag.

**Queried Object OD Version Number**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient le numéro de version de l'OD de l'objet interrogé. Il est présent pour toutes les interrogations, à l'exception des interrogations Physical Device Tag.

**Queried Object Numeric Id**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient l'attribut Numeric ID (identificateur numérique) ou OD Index (indice d'OD) de l'objet interrogé. Il est présent si l'objet interrogé est contenu dans un VFD.

**Duplicate Detection State**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**List of VCR References**

Ce paramètre de type conditionnel, lorsqu'il est présent, contient une liste de références de VCR qui peuvent être utilisées avec le service Initiate de Type 9 pour ouvrir une VCR vers l'objet interrogé. Il est présent pour toutes les interrogations, à l'exception des interrogations Physical Device Tag.

#### 7.2.2.3.3.3 Procédure du service

La procédure de service non confirmé spécifiée à l'Article 4 s'applique à ce service.

### 7.2.2.3.4 Service Identify

#### 7.2.2.3.4.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour obtenir des informations d'identification issues d'un SMK distant.

#### 7.2.2.3.4.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 112.

**Tableau 112 – Paramètres du service Identify**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| SMK ID | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | M | M (=) |
| Device ID | | | M | M (=) |
| Physical Device Tag | | | M | M (=) |
| SMK State | | | M | M (=) |
| Device Type | | | M | M (=) |
| LAN Redundancy Socket Address | | | C | C (=) |
| Annunciation Repeat Time | | | C | C (=) |
| Device Redundancy State | | | C | C (=) |
| Duplicate Detection State | | | C | C (=) |
| Device Index | | | C | C (=) |
| Max Device Index | | | C | C (=) |
| Operational Network Address | | | C | C (=) |
| SMK Attributes Version Number | | | C | C (=) |
| Device Version Number | | | C | C (=) |
| Version Number List | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la demande a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande est envoyée.

**SMK ID**

Ce paramètre identifie le SMK destiné à répondre à la demande. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'une interface de liaison de Type "X" dans un appareil de liaison de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**SMK ID**

Ce paramètre identifie le SMK envoyant la réponse. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'une interface de liaison de Type "X" dans un appareil de liaison de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Device ID; Identificateur d'appareil**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Physical Device Tag (Étiquette d'appareil physique)**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**SMK State**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Device Type**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**LAN Redundancy Socket Address**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Annunciation Repeat Time**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Device Redundancy State**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Duplicate Detection State**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Device Index**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Max Device Index**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Operational Network Address**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**SMK Attributes Version Number**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Device Version Number**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si le répondeur est un appareil de Type 5.

**Version Number List**

Ce paramètre conditionnel est utilisé par l'appareil qui connecte ensemble des liaisons de deux ou plusieurs types de modèle de communications. Il est présent si le répondeur est un appareil de Type 5.

Si le paramètre de demande d'adresse de Type "X" n'est pas présent, ce paramètre spécifie l'identificateur de liaison (Link ID) et le numéro de version (Version Number) de chaque liaison attachée. Ce numéro de version de liaison de Type "X" est incrémenté chaque fois qu'une adresse d'appareil est ajoutée au réseau de Type "X" ou en est supprimée.

Si le paramètre de demande d'adresse de Type "X" identifie une liaison spécifique connectée à l'appareil récepteur, ce paramètre spécifie l'adresse de Type "X" et le numéro de version associés à chaque appareil connecté à la liaison en question. Ce numéro de version d'adresse de l'appareil de Type "X" est incrémenté chaque fois qu'une adresse spécifique d'appareil est ajoutée au réseau de Type "X" ou en est supprimée.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**SMK ID**

Ce paramètre identifie le SMK envoyant la réponse. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'une interface de liaison de Type "X" dans un appareil de liaison de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

#### 7.2.2.3.4.3 Procédure de service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service.

#### 7.2.2.3.5 Service Device annunciation

##### 7.2.2.3.5.1 Vue d'ensemble du service

Ce service non confirmé est utilisé pour multidiffuser une notification sur le réseau qui contient des informations de base relatives à un SMK. L'adresse de multidiffusion Annunciation réservée est utilisée.

##### 7.2.2.3.5.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 113.

**Tableau 113 – Paramètres du service Annunciate**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| Source Address | | M |
| SMK ID | M | M (=) |
| Device ID | M | M (=) |
| Physical Device Tag | M | M (=) |
| SMK State | M | M (=) |
| Device Type | M | M (=) |
| LAN Redundancy Socket Address | M | M (=) |
| Annunciation Repeat Time | M | M (=) |
| Device Redundancy State | M | M (=) |
| Duplicate Detection State | M | M (=) |
| Device Index | M | M (=) |
| Max Device Index | M | M (=) |
| Operational Network Address | M | M (=) |
| SMK Attributes Version Number | M | M (=) |
| Device Version Number | M | M (=) |
| Version Number List | C | C (=) |

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la demande a été envoyée.

**Device ID**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Physical Device Tag**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Device Type**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**LAN Redundancy Addr**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Annunciation Repeat Time**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**SMK State Capability**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Device Redundancy State**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Duplicate Detection State**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Operational Network Address**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Device Index**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Max Device Index**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**SMK Attributes Version Number**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Device Version Number**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom.

**Version Number List**

Ce paramètre conditionnel est présent si le service a été demandé par un appareil qui connecte ensemble des liaisons de deux ou plusieurs types de Communication Model Types et si le paramètre de demande d'adresse de Type "X" n'est pas présent. Ce paramètre spécifie le numéro de version de chaque liaison attachée.

#### 7.2.2.3.5.3 Procédure du service

La procédure de service non confirmé spécifiée à l'Article 4 s'applique à ce service.

### 7.2.2.3.6 Service Set assignment info

#### 7.2.2.3.6.1 Vue d'ensemble du service

Ce service est utilisé pour mettre à jour les valeurs d'attributs du SMK adressé par le paramètre Destination Address. Le SMK recevant les nouvelles valeurs met à jour ses attributs, si possible, et retourne une réponse qui inclut les valeurs d'attributs nouvellement mises à jour. Ce service peut aussi être invoqué pour attribuer un nouveau PD Tag et une nouvelle adresse à un appareil de Type "X" qui est connecté à un appareil de Type 5 à travers un réseau de Type "X".

#### 7.2.2.3.6.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 114.

**Tableau 114 – Paramètres du service Set assignment info**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| SMK ID | M | M (=) | | |
| Device ID | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| LAN Redundancy Socket Address | C | C (=) | | |
| Device Redundancy State | C | C (=) | | |
| Clear Duplicate Detection State | C | C (=) | | |
| Operational Network Address | C | C (=) | | |
| Device Index | C | C (=) | | |
| Max Device Index | C | C (=) | | |
| Annunciation Repeat Time | C | C (=) | | |
| New Type "X" Address | C | C (=) | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Annunciation Repeat Time | | | C | C (=) |
| Max Device Index | | | C | C (=) |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Error Info | | | M | M (=) |

NOTE Voir la Note en 3.8.4.3.

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle le service a été demandé.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande est envoyée.

**SMK ID**

Ce paramètre identifie le SMK destiné à répondre à la demande. Le répondeur peut être:

– un SMK de Type 5, ou

– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Device ID; Identificateur d'appareil**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom. Il convient que cette valeur concorde avec celle de l'attribut de SMK et elle n'est pas inscriptible.

**Physical Device Tag**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom. Cette valeur est inscriptible, mais seulement lorsque l'appareil n'a pas de Physical Device Tag. Les Physical Device Tag peuvent être dégagés au moyen du service Clear Assignment Info.

**LAN Redundancy Socket Address**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Cette valeur est inscriptible. Il est présent si la demande est adressée à un SMK de Type 5.

**Device Redundancy State**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Cette valeur est inscriptible. Il est présent si la demande est adressée à un SMK de Type 5.

**Clear Duplicate Detection State**

Ce paramètre conditionnel est utilisé pour mettre la valeur de l'attribut Duplicate Detection State du SMK à "Duplicate Physical Device Tag Not Detected" et/ou à "Duplicate Device Index Not Detected". Il est présent si la demande est adressée à un SMK de Type 5.

**Operational Network Address**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Cette valeur est inscriptible. Il est présent si la demande est adressée à un SMK de Type 5.

**Device Index**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si la demande est adressée à un SMK de Type 5.

**Max Device Index**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il est présent si la demande est adressée à un SMK de Type 5. Le répondeur peut négocier cette valeur à une valeur plus faible, mais pas inférieure à 500. Autrement dit, une valeur inférieure à 500 ne peut être retournée que si la valeur en question ou une valeur inférieure est reçue dans la demande.

**Annunciation Repeat Time**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Cette valeur est inscriptible. Il est sélectionné si la demande est adressée à un SMK de Type 5. Au cours de l'affectation des appareils, l'agent FDA peut négocier cette valeur à une valeur plus grande, pas supérieure à 5000, si la valeur attribuée est inférieure à 5000 et l'appareil n'est pas capable de prendre en charge la valeur attribuée.

**New Type "X" Address**

Ce paramètre conditionnel spécifie une valeur NULL ou une adresse à affecter à un appareil de Type "X" connecté à un appareil de Type 5 à travers un réseau de Type "X". Il est présent si la demande est adressée à un SMK de Type “X”.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**SMK ID**

Ce paramètre identifie le SMK envoyant la réponse. Le répondeur peut être

– un SMK de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Annunciation Repeat Time**

Ce paramètre conditionnel spécifie la valeur négociée de l'attribut de SMK du même nom. Il est présent si le service a été demandé à partir d'un SMK d'appareil de Type 5.

**Max Device Index**

Ce paramètre conditionnel spécifie la valeur négociée de l'attribut de SMK du même nom. Il est présent si le service a été demandé à partir d'un SMK d'appareil de Type 5.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Type "X" Address**

Ce paramètre conditionnel spécifie l'adresse de Type X à partir de laquelle le service a été demandé. Il peut s'agir de l'adresse de la liaison de Type "X" connectée à un appareil de Type 5 ou de l'adresse d'un appareil sur une liaison de Type "X" qui est connecté à un appareil de Type 5 à partir duquel le service a été demandé. Il est présent si le service a été demandé à partir d'un d'appareil de Type "X".

#### 7.2.2.3.6.3 Procédure du service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service. Le service échoue si les paramètres Device ID et Device Type ne concordent pas avec ceux du SMK adressé.

### 7.2.2.3.7 Service Clear assignment info

#### 7.2.2.3.7.1 Vue d'ensemble du service

Ce service est utilisé pour supprimer un appareil en supprimant les informations d'attribution de SMK et de redondance LAN et en mettant tous les VFD aux valeurs d'usine. Il peut être utilisé pour supprimer le PD Tag d'un appareil sur une liaison de Type "X" qui est connecté à un appareil de Type 5. Si le SMK ID indique qu'il faut qu'un PD Tag de Type "X" soit supprimé, le SMK invoque les procédures appropriées du SMK de Type "X" SM pour le supprimer.

L'application de configuration inclut le PD Tag (qui peut être vide) et le Device ID pour s'assurer que le bon appareil recevra ce service. Si le PD Tag ou bien le Device ID ne concorde pas, l'appareil retourne une réponse négative.

#### 7.2.2.3.7.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 115.

**Tableau 115 – Paramètres du service Clear assignment info**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| SMK ID | M | M (=) | | |
| Device ID | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle le service a été demandé.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande est envoyée.

**SMK ID**

Ce paramètre identifie le SMK destiné à répondre à la demande. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Device ID; Identificateur d'appareil**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom. Il convient que cette valeur concorde avec celle de l'attribut de SMK en question.

**Physical Device Tag**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom. Il convient que cette valeur concorde avec celle de l'attribut de SMK en question.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**SMK ID**

Ce paramètre identifie le SMK envoyant la réponse. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**SMK ID**

Ce paramètre identifie le SMK envoyant la réponse. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

#### 7.2.2.3.7.3 Procédure du service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service. Le service échoue si les paramètres Device ID et Device Type ne concordent pas avec ceux du SMK adressé.

### 7.2.2.3.8 Service Clear address

#### 7.2.2.3.8.1 Vue d'ensemble du service

Ce service est utilisé pour supprimer l'adresse d'un appareil de Type 5, d'une interface de Type "X" connectée à un appareil de Type 5 ou d'un appareil sur une liaison de Type "X" qui est connecté à un appareil de Type 5. Après suppression de l'adresse d'une interface de Type "X" connectée à un appareil de Type 5 ou d'un appareil sur une liaison de Type "X" qui est connecté à un appareil de Type 5, le SMK de l'appareil répondeur retourne la réponse. Après avoir accepté la demande de supprimer sa propre adresse, le SMK d'appareil retourne la réponse puis supprime son adresse.

Ce service opère sur des sessions client/serveur établies en dynamique ou préétablies.

#### 7.2.2.3.8.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 116.

**Tableau 116 – Paramètres du service Clear address**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| AREP | M | M | | |
| Invoke ID | U | | | |
| Destination Address | M | M (=) | | |
| SMK ID | C | C (=) | | |
| Device ID | M | M (=) | | |
| Physical Device Tag | M | M (=) | | |
| InterfaceToClear | C | C (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Destination Address | | | M | M (=) |
| SMK ID | | | C | C (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Destination Address | | | M | M (=) |
| SMK ID | | | M | M (=) |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**Destination Address**

Ce paramètre est l'adresse à laquelle la demande de service est à envoyer. La demande peut être adressée à un appareil de Type 5, à une liaison de Type "X" connectée à un appareil de Type 5 ou à un appareil sur une liaison de Type "X" qui est connecté à un appareil de Type 5.

**SMK ID**

Ce paramètre identifie le SMK destiné à répondre à la demande. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Device ID; Identificateur d'appareil**

Ce paramètre spécifie la valeur de l'attribut de SMK du même nom. Il convient que cette valeur concorde avec celle de l'attribut de SMK en question.

**Physical Device Tag (Étiquette d'appareil physique)**

Ce paramètre conditionnel spécifie la valeur de l'attribut de SMK du même nom. Il convient que cette valeur concorde avec celle de l'attribut de SMK en question.

**InterfaceToClear**

Ce paramètre indique lesquelles des interfaces réseau de l'appareil sont à supprimer. Il est présent si la demande est adressée à un SMK de Type 5.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Destination Address**

Ce paramètre est l'adresse à laquelle la réponse de service est à retourner.

**SMK ID**

Ce paramètre identifie le SMK qui répond à la demande. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Destination Address**

Ce paramètre est l'adresse à laquelle la réponse de service d'erreur est à retourner.

**SMK ID**

Ce paramètre identifie le SMK qui répond à la demande. Le répondeur peut être:

– un SMK de Type 5, ou
– un SMK de Type "X" d'un appareil de Type "X" connecté à un appareil de liaison de Type 5.

#### 7.2.2.3.8.3 Procédure de service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service.

En outre, il convient que les valeurs des paramètres Device ID, Physical Device Tag et Device Type concordent avec les valeurs d'attributs correspondantes dans le SMK de l'appareil dont la demande d'adresse est à supprimer. Si elles ne concordent pas, une réponse négative est retournée.

### 7.2.3 ASE LAN Redundancy

#### 7.2.3.1 Vue d'ensemble

Les appareils peuvent avoir deux interfaces réseau qui sont utilisées pour la redondance LAN. Ces interfaces sont appelées Interface réseau A (Network Interface A) et Interface réseau B (Network Interface B). Elles sont gérées par un utilisateur de l'ASE "LAN Redundancy" appelé " LAN Redundancy Entity" (Entité de redondance LAN).

LAN Redundancy Entity échange des informations de statut de redondance avec d'autres entités de redondance LAN en utilisant le service Diagnostic Message. LAN Redundancy Entity d'un appareil sélectionne l'interface réseau utilisée pour envoyer d'autres APDU en fonction de ces informations de statut.

Chaque LAN Redundancy Entity présente une primitive "request" de service Diagnostic Message à intervalles réguliers et l'ASE LAN Redundancy construit deux APDU de service Diagnostic Message et les envoie simultanément sur l'interface réseau A et l'Interface réseau B. Chaque LAN Redundancy Entity reçoit des paires d'indications de service issues des autres appareils sur le réseau qui participent à la redondance LAN. Ces informations sont utilisées pour déterminer l'interface réseau (A ou B) qui est à sélectionner pour envoyer d'autres APDU.

#### 7.2.3.2 Spécification de la classe LAN redundancy

##### 7.2.3.2.1 Modèle formel de LAN redundancy

**FAL ASE:** **LAN Redundancy**
**CLASS:** **LAN Redundancy**
**CLASS ID:** non utilisé
**PARENT CLASS:** TOP
**ATTRIBUTES:**

| | | | |
|---|---|---|---|
| 1 | (m) | Attribut: | LAN Redundancy Attributes Version |
| 2 | (m) | Attribut: | Number of Network Interfaces |
| 3 | (m) | Attribut: | Max Sequence Number Difference |
| 4 | (m) | Attribut: | LAN Redundancy Flags |
| 5 | (m) | Attribut: | Diagnostic Message Interval |
| 6 | (m) | Attribut: | Aging Time |
| 7 | (m) | Attribut: | Diagnostic Message Send Addresses |
| 8 | (m) | Attribut: | Diagnostic Message Receive Addresses |
| 9 | (m) | Attribut: | List of Network Interface Status |
| 10 | (m) | Attribut: | Number of Diagnostic Service Indications Received |
| 11 | (m) | Attribut: | Number of Diagnostic Message Indications Missed |
| 12 | (m) | Attribut: | Number of Faults Detected |
| 13 | (m) | Contrainte: | Crossed Cable Detection Enabled = TRUE |
| 13.1 | (m) | Attribut: | Number of Diagnostic Service Indications Missed |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1 | (m) | OpsService: | Diagnostic Message |
| 2 | (o) | OpsService: | Get Redundancy Information |
| 3 | (o) | OpsService: | Put Redundancy Information |
| 4 | (o) | OpsService: | Get Redundancy Statistics |

##### 7.2.3.2.2 Attributs

**LAN Redundancy Attributes Version**

Cet attribut spécifie la version de cet objet. Chaque fois que la valeur de l'un quelconque de ses autres attributs change, le numéro de version est incrémenté de 1. Il survit à une panne d'alimentation si le contenu d'APDU survit à la panne d'alimentation; autrement, sa valeur est remise à 0. C'est la seule fois que sa valeur est remise à 0. Le numéro de version 0 indique que cet objet n'a pas été configuré.

**Number of Network Interfaces**

Ce paramètre spécifie le nombre d'interfaces réseau sur cet appareil. Un appareil peut avoir une ou deux interfaces réseau. Celles-ci sont étiquetées Interface réseau A et Interface réseau B. (L'interface réseau B n'est utilisée que s'il y a deux interfaces réseau.)

**Max Sequence Number Difference**

Cet attribut définit la différence maximale admissible entre les paramètres "numéro de séquence" dans une paire d'indications de service Diagnostic reçues en provenance d'un seul et même appareil expéditeur. Lorsque la différence excède la valeur de cet attribut, il est détecté un défaut dans le trajet partant de l'interface réseau qui envoie le numéro de séquence le plus faible.

**LAN Redundancy Flags**

Cet attribut commande la façon dont le service Diagnostic Message est utilisé par l'Entité de redondance LAN. Sa valeur peut indiquer un ou plusieurs des composants suivants:

– Single Multicast APDU Transmission Interface Enabled (Unique Interface d'émission d'APDU en multidiffusion activée). La politique d'émission pour tous les services avec des

adresses de destination en multidiffusion, à l'exception des services Diagnostic Message et SMK Annunciate.

False Émission sur les deux interfaces réseau

True Émission sur une seule interface réseau

– Crossed Cable Detection Enabled (Détection de câbles croisés activée).

False Ne pas détecter de câbles croisés

True Détecter des câbles croisés

– Single Multicast APDU Reception Interface Enabled (Unique Interface de réception d'APDU en multidiffusion activée). La politique de réception pour toutes les adresses de destination en multidiffusion, à l'exception de l'adresse SM Multicast (SM en multidiffusion) et des LAN Diagnostic Message Receive Adresses (adresses de réception de message de diagnostic LAN) pour les interfaces A et B.

False Se mettre à l'écoute des adresses de multidiffusion sur les deux interfaces réseau

True Se mettre à l'écoute des adresses de multidiffusion sur une seule interface réseau s'il est détecté zéro ou une seule panne.

– Diagnosis using own APDUs Enabled. (Diagnostic utilisant des APDU propres activé)

False Pour le diagnostic, ne pas utiliser de messages de diagnostic propres.

True Pour le diagnostic, utiliser des messages de diagnostic propres

– Load Balancing Enabled (Équilibrage de charge activé).

False Ne pas effectuer d'équilibrage de charge

True Effectuer l'équilibrage de charge

**Diagnostic Message Interval**

Cet attribut spécifie l'Intervalle de temps en millisecondes entre des invocations successives du service Diagnostic Message par l'Entité de redondance LAN.

**Aging Time**

Cet attribut spécifie l'Intervalle de temps utilisé par l'Entité de redondance LAN pour retirer les appareils silencieux de sa liste d'appareils qui participent à la redondance LAN. Il convient que la valeur de cet attribut soit plus grande que la valeur de l'attribut *Max Sequence Number Difference* multipliée par la plus grande valeur du paramètre *Diagnostic Message Interval* trouvé dans les indications de service Diagnostic Message.

**Diagnostic Message Send Addresses**

Cet attribut définit les adresses de destination pour envoyer les APDU Diagnostic Message. Une seule adresse est définie pour chaque interface réseau à partir de laquelle les APDU Diagnostic Message sont envoyées. Les adresses peuvent être des adresses de diffusion, de multidiffusion ou de monodiffusion.

**Diagnostic Message Receive Addresses**

Cet attribut définit les adresses pour recevoir les APDU Diagnostic Message. Une seule adresse est définie pour chaque interface réseau sur laquelle les APDU Diagnostic Message sont reçues. Les adresses peuvent être des adresses de diffusion, de multidiffusion ou de monodiffusion.

**List of Network Interface Status**

Cet attribut est une liste. Chaque entrée dans cette liste représente le statut pour le trajet allant d'un appareil distant à partir duquel les indications de service Diagnostic Message peuvent être reçues. Le statut est bidirectionnel pour le trajet et indique si, oui ou non, les indications de service Diagnostic Message sont reçues avec succès à chaque extrémité du trajet ou si, oui ou non, l'une ou l'autre extrémité du trajet est à l'état de panne (une panne a été détectée, mais n'a pas été éliminée).

**Number of Diagnostic Message Service Indications Received**

Cet attribut compte le nombre d'indications de service Diagnostic Message reçues (en provenance de tous les appareils). Un compte distinct est maintenu pour chaque interface réseau.

**Number of Diagnostic Message Service Indications Missed**

Cet attribut est un compte du nombre d'indications de service Diagnostic Message manquées (en provenance de tous les appareils) tel que calculé à partir des trous dans les numéros de séquence. Un compte distinct est maintenu pour chaque interface réseau.

**Number of Faults Detected**

Cet attribut est un compte du nombre total de pannes détectées à partir d'indication de service Diagnostic Message. Un compte distinct est maintenu pour chaque interface réseau.

**List of Crossed Cable Status**

Cet attribut conditionnel est une liste. Cet attribut n'est présent que si la valeur du composant *Crossed Cable Detection Enabled* de l'attribut LAN Redundancy Flags est TRUE. Chaque entrée dans cette liste représente le statut de câbles croisés pour le trajet allant vers un appareil distant à partir duquel les indications de service Diagnostic Messages peuvent être reçues. Le statut de câbles croisés indique si, oui ou non, l'interface réseau utilisée par le service Diagnostic Message pour les demandes, telle qu'indiquée par le paramètre *Network Interface used for Transmission of this Diagnostic Message* dans le service Diagnostic Message, est la même interface réseau utilisée par l'appareil rapporteur pour recevoir les messages.

#### 7.2.3.2.3 Services

**Diagnostic Message**

Ce service est utilisé pour envoyer simultanément le même message de diagnostic à partir de chaque interface réseau.

**Get Redundancy Information**

Ce service est utilisé pour récupérer les attributs de LAN Redundancy.

**Set Redundancy Information**

Ce service est utilisé pour mettre à jour les attributs de LAN Redundancy.

**Get Redundancy Statistics**

Ce service est utilisé pour récupérer les statistiques de LAN Redundancy.

### 7.2.3.3 Spécification des services de l'ASE LAN Redundancy

#### 7.2.3.3.1 Vue d'ensemble

Ce paragraphe contient la définition des services qui est propre à cet ASE. Les services de LAN Redundancy opèrent directement sur les services de sockets. Ils n'utilisent pas d'AR.

Les services définis pour cet ASE sont:

Diagnostic Message
Get Redundancy Info
Put Redundancy Info
Get Redundancy Statistics

#### 7.2.3.3.2 Service Diagnostic message

##### 7.2.3.3.2.1 Vue d'ensemble du service

Ce service est utilisé pour faire envoyer simultanément des APDU de service Diagnostic Message à partir de chacune des interfaces de réseau redondant, étiquetée Interface A et Interface B. Les messages sont identiques, à l'exception de l'identification de l'interface qui avait été utilisée pour envoyer chaque APDU.

##### 7.2.3.3.2.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 117.

**Tableau 117 – Service Diagnostic message**

| Nom de paramètre | Req | Ind |
|---|---|---|
| Argument | | |
| Source Address | | M |
| Destination Address | M | M (=) |
| Device Index | M | M (=) |
| Number of Interfaces | M | M (=) |
| Transmission Interface | M | M (=) |
| Diagnostic Message Interval | M | M (=) |
| SMK PD Tag | M | M (=) |
| SMK Duplicate Detection State | M | M (=) |
| List of Network Interface Status | M | M (=) |

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la demande a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande avait été envoyée.

**Device Index**

Cette valeur est obtenue à partir de l'attribut de SMK du même nom.

**Number of Network Interfaces**

Cette valeur est dérivée de l'attribut List of Network Addresses du SMK. Il y a une adresse réseau par interface réseau dans la liste.

**Transmission Interface**

Ce paramètre est l'interface d'émission à partir de laquelle la demande avait été envoyée.

**Diagnostic Message Interval**

Cette valeur est obtenue à partir de l'attribut de LAN Redundancy du même nom.

**SMK PD Tag**

Ce paramètre spécifie la valeur de l'attribut PD Tag de SMK de l'appareil.

**SMK Duplicate Detection State**

Ce paramètre spécifie la valeur de l'attribut Duplicate Detection State de SMK de l'appareil.

**List of Network Interface Status**

Cette valeur est obtenue à partir de l'attribut de LAN Redundancy du même nom.

#### 7.2.3.3.2.3 Procédure du service

La procédure de service non confirmé spécifiée à l'Article 4 s'applique à ce service.

### 7.2.3.3.3 Service Get redundancy info

#### 7.2.3.3.3.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour récupérer les attributs de LAN Redundancy.

#### 7.2.3.3.3.2 Paramètres du service

Les paramètres de service pour ce service sont montrés dans le Tableau 118.

**Tableau 118 – Service Get redundancy info**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| | | | | |
| LAN Redundancy Attributes Version | | | M | M (=) |
| Number of Network Interfaces | | | M | M (=) |
| Max Sequence Number Difference | | | M | M (=) |
| LAN Redundancy Flags | | | M | M (=) |
| Diagnostic Message Interval | | | M | M (=) |
| AgingTime | | | M | M (=) |
| Diagnostic Message Interface Send Addresses | | | M | M (=) |
| Diagnostic Message Interface Receive Addresses | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la demande a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande avait été envoyée.

**LAN Redundancy Attributes Version**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Number of Network Interfaces**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Max Sequence Number Difference**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**LAN Redundancy Flags**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interval**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**AgingTime**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interface Send Addresses**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interface Receive Addresses**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

#### 7.2.3.3.3.3 Procédure du service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service.

#### 7.2.3.3.4 Service Put redundancy info

##### 7.2.3.3.4.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour récupérer les attributs de LAN Redundancy. Après mise à jour des attributs contenus dans la demande, le répondeur retourne les valeurs et le numéro de version mis à jour des attributs de LAN Redundancy.

##### 7.2.3.3.4.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 119.

**Tableau 119 – Service Put redundancy info**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| | | | | |
| LAN Redundancy Attributes Version | | | M | M (=) |
| Number of Network Interfaces | | | M | M (=) |
| Max Sequence Number Difference | | | M | M (=) |
| LAN Redundancy Flags | | | M | M (=) |
| Diagnostic Message Interval | | | M | M (=) |
| AgingTime | | | M | M (=) |
| Diagnostic Message Interface Send Addresses | | | M | M (=) |
| Diagnostic Message Interface Receive Addresses | | | M | M (=) |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| | | | | |
| LAN Redundancy Attributes Version | | | M | M (=) |
| Number of Network Interfaces | | | M | M (=) |
| Max Sequence Number Difference | | | M | M (=) |
| LAN Redundancy Flags | | | M | M (=) |
| Diagnostic Message Interval | | | M | M (=) |
| AgingTime | | | M | M (=) |
| Diagnostic Message Interface Send Addresses | | | M | M (=) |
| Diagnostic Message Interface Receive Addresses | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la demande a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande avait été envoyée.

**LAN Redundancy Attributes Version**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Number of Network Interfaces**

Cette valeur contient la nouvelle valeur de l'attribut de LAN Redundancy du même nom.

**Max Sequence Number Difference**

Cette valeur contient la nouvelle valeur de l'attribut de LAN Redundancy du même nom.

**LAN Redundancy Flags**

Cette valeur contient la nouvelle valeur de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interval**

Cette valeur contient la nouvelle valeur de l'attribut de LAN Redundancy du même nom.

**AgingTime**

Cette valeur contient la nouvelle valeur de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interface Send Addresses**

Cette valeur contient la nouvelle valeur de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interface Receive Addresses**

Cette valeur contient la nouvelle valeur de l'attribut de LAN Redundancy du même nom.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**LAN Redundancy Attributes Version**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**Number of Network Interfaces**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**Max Sequence Number Difference**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**LAN Redundancy Flags**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interval**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**AgingTime**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interface Send Addresses**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**Diagnostic Message Interface Receive Addresses**

Cette valeur contient la valeur mise à jour de l'attribut de LAN Redundancy du même nom.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

#### 7.2.3.3.4.3 Procédure du service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service.

### 7.2.3.3.5 Service Get redundancy statistics

#### 7.2.3.3.5.1 Vue d'ensemble du service

Ce service confirmé est utilisé pour récupérer les attributs de statistiques de LAN Redundancy.

#### 7.2.3.3.5.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 120.

**Tableau 120 – Service Get redundancy statistics**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Number of Diagnostic Message Service Indications Received | | | M | M (=) |
| Number of Diagnostic Message Service Indications Missed | | | M | M (=) |
| Number of Faults Detected | | | M | M (=) |
| List of Crossed Cable Status | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la demande a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande avait été envoyée.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**Number of Diagnostic Message Service Indications Received**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Number of Diagnostic Message Service Indications Missed**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Number of Faults Detected**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**List of Crossed Cable Status**

Cette valeur contient la valeur de l'attribut de LAN Redundancy du même nom.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

#### 7.2.3.3.5.3 Procédure du service

La procédure de service confirmé spécifiée à l'Article 4 s'applique à ce service.

## 7.3 Sessions FDA

### 7.3.1 Spécification de la classe Publisher/subscriber session endpoint

#### 7.3.1.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge la distribution en file d'attente à la demande de services non confirmés vers un ou plusieurs processus application. Il s'agit de la contrepartie de Type 5 des AREP BNU du Type 9. Le comportement de ce type d'AR peut être décrit comme suit.

Un utilisateur ASE AR souhaitant acheminer une APDU de service non confirmé présente une unité de données de service d'ASE AR au point d'extrémité expéditeur de l'AR. L'AREP qui envoie l'APDU de demande la présente à la couche sous-jacente en vue de son transfert. La couche sous-jacente l'envoie à la prochaine occasion. L'AREP qui reçoit l'APDU de demande provenant de sa couche sous-jacente la livre aux utilisateurs sélectionnés d'ASE AR dans l'ordre dans lequel elle a été reçue. Les utilisateurs ASE AR sont sélectionnés en fonction des informations d'adresse contenues dans l'en-tête AR.

Le point d'extrémité expéditeur a un rôle de PUBLISHER alors que les points d'extrémité récepteurs ont un rôle de SUBSCRIBER.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | PUBLISHER<br>SUBSCRIBER |
| Cardinalité: | 1 à n |
| Cohérence temporelle: | Non |

#### 7.3.1.2 Modèle formel

| | |
|---|---|
| **FAL ASE:** | **ASE AR** |
| **CLASS:** | **Publisher/Subscriber Session Endpoint** |
| **CLASS ID:** | **Non utilisé** |
| **PARENT CLASS:** | **TOP** |

**ATTRIBUTS DE GESTION DE RÉSEAU**

| | | | |
|---|---|---|---|
| 1. | (m) | Attribut: | Role (PUBLISHER, SUBSCRIBER) |

2. (m) Attribut: AREP State (OPEN, CLOSED)
3. (m) Attribut: Socket Mapping Reference

**SERVICES:**

1. (o) OpsService: Unconfirmed Send

#### 7.3.1.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

PUBLISHER Des points d'extrémité de ce type envoient des messages vers plusieurs points d'extrémité en utilisant des services non confirmés opérant sur des services de liaison de données sans connexion.

SUBSCRIBER Des points d'extrémité de ce type reçoivent des rapports issus de points d'extrémité sources en utilisant des services non confirmés opérant sur des services de socket sans connexion.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Socket Mapping Reference**

Cet attribut est une référence au mapping du socket à la couche sous-jacente. Ces mappings sont définis dans la CEI 61158-6-5.

#### 7.3.1.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé. Ce service est spécifié par le Type 9.

### 7.3.2 Spécification de la classe Report distribution session endpoint

#### 7.3.2.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge la distribution en file d'attente, à la demande, de services non confirmés vers un ou plusieurs processus application. Il s'agit de la contrepartie du Type 5 des AREP QUU du Type 9. Le comportement de ce type d'AR peut être décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de service non confirmé présente une unité de données de service ASE AR au point d'extrémité émettrice de l'AR. L'AREP qui envoie l'APDU de demande la présente à la couche sous-jacente en vue de son transfert. La couche sous-jacente l'envoie à sa prochaine occasion. L'AREP qui reçoit l'APDU de demande provenant de sa couche sous-jacente la livre à tous les utilisateurs ASE AR de l'AREP destinataire dans l'ordre dans lequel elle a été reçue.

Le point d'extrémité expéditeur a un rôle de REPORT SOURCE alors que les points d'extrémité récepteurs ont un rôle de REPORT SINK.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | REPORT SOURCE<br>REPORT SINK |
| Cardinalité: | 1 à n |
| Cohérence temporelle: | Non |

#### 7.3.2.2 Modèle formel

| | | | |
|---|---|---|---|
| **FAL ASE:** | | | **ASE AR** |
| **CLASS:** | | | **Report Distribution Session Endpoint** |
| **CLASS ID:** | | | **Non utilisé** |
| **PARENT CLASS:** | | | **TOP** |
| **ATTRIBUTS DE GESTION DE RÉSEAU** | | | |
| 1. | (m) | Attribut: | Role (REPORT SOURCE, REPORT SINK) |
| 2. | (m) | Attribut: | AREP State (OPEN, CLOSED) |
| 3. | (m) | Attribut: | Socket Mapping Reference |
| **SERVICES:** | | | |
| 1. | (o) | OpsService: | Unconfirmed Send |

#### 7.3.2.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

REPORT SOURCE Des points d'extrémité de ce type envoient des messages vers plusieurs points d'extrémité REPORT SINK en utilisant des services non confirmés opérant sur des services de socket sans connexion.

REPORT SINK Des points d'extrémité de ce type reçoivent des rapports issus de points d'extrémité REPORT SOURCE en utilisant des services non confirmés opérant sans connexion.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Socket Mapping Reference**

Cet attribut est une référence au mapping du socket sur la couche sous-jacente. Ces mappings sont définis dans la CEI 61158-6-5.

#### 7.3.2.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé. Ce service est spécifié par le Type 9.

### 7.3.3 Spécification de la classe Client/server AR endpoint

#### 7.3.3.1 Vue d'ensemble de la classe

Cette classe est définie pour prendre en charge l'échange, à la demande, de services confirmés et non confirmés entre deux processus application. Il s'agit de la contrepartie de Type 5 des AREP QUB-CO du Type 9. Le comportement de ce type d'AR peut être décrit comme suit.

Un utilisateur d'ASE AR souhaitant acheminer une APDU de demande ou de réponse la présente à son AREP comme une unité de données de service ASE AR. L'AREP qui envoie l'APDU de demande la met en file d'attente à la couche sous-jacente en vue de son transfert à la prochaine occasion. L'un ou l'autre des points d'extrémité peut envoyer des APDU de demande ou de réponse de service confirmé. Seul l'AREP Client peut envoyer l'APDU de demande de service OpenSession APDU.

L'AREP qui reçoit l'APDU provenant de sa couche sous-jacente la met en file d'attente en vue de sa distribution à son utilisateur d'ASE AR dans l'ordre dans lequel elle a été reçue.

Les caractéristiques de cette classe d'AREP sont résumées ci-après.

| | |
|---|---|
| Rôles: | Client<br>Server |
| Cardinalité: | 1 à 1 |
| Cohérence temporelle: | Non |

#### 7.3.3.2 Modèle formel

**FAL ASE:** **ASE AR**
**CLASS:** **Client/server AREP**
**CLASS ID:** **Non utilisé**
**PARENT CLASS:** **TOP**
**ATTRIBUTS DE GESTION DE RÉSEAU**

| | | | |
|---|---|---|---|
| 1. | (m) | Attribut: | Role (CLIENT, SERVER) |
| 2. | (m) | Attribut: | AREP State |
| 3. | (m) | Attribut: | Server Physical Device Tag |
| 4. | (m) | Attribut: | Socket Mapping Reference |

**SERVICES:**

| | | | |
|---|---|---|---|
| 1. | (o) | OpsService: | Confirmed Send |
| 2. | (o) | OpsService: | Unconfirmed Send |
| 3. | (o) | OpsService: | OpenSession |

#### 7.3.3.3 Attributs de gestion de réseau

**Role**

Cet attribut spécifie le rôle de l'AREP. Les valeurs valides sont:

CLIENT Des points d'extrémité de ce type émettent des APDU "request" de services confirmés pour initier l'établissement de l'AR. Il convient qu'un et un seul des AREP impliqués dans une AR soit l'initiateur.

SERVER Des points d'extrémité de ce type répondent à des demandes reçues en provenance du CLIENT d'établir l'AR.

**AREP State**

Cet attribut spécifie l'état de l'AREP. Les valeurs pour cet attribut sont spécifiées dans la CEI 61158-6-5.

**Server Physical Device Tag**

Cet attribut conditionnel spécifie l'attribut Physical Device Tag de SMK de l'appareil serveur. Cet attribut est utilisé pour localiser le serveur sur le réseau. La façon dont le serveur est situé ne relève pas du domaine d'application de la présente norme. Cet attribut est présent uniquement pour les AREP de type CLIENT.

**Socket Mapping Reference**

Cet attribut est une référence au mapping du socket à la couche sous-jacente. Ces mappings sont définis dans la CEI 61158-6-5.

#### 7.3.3.4 Services

**Unconfirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service non confirmé. Ce service est spécifié par le Type 9.

**Confirmed Send**

Ce service facultatif est utilisé pour envoyer sur une AR un service confirmé. Ce service est spécifié par le Type 9.

**OpenSession**

Ce service est utilisé pour établir une AR.

### 7.3.4 Spécification des services de l'ASE FDA Session

#### 7.3.4.1 Services pris en charge

Ce paragraphe contient la définition des services qui est propre à cet ASE.

Les services définis pour cet ASE sont:

Service Open session
Service Idle

#### 7.3.4.2 Service Open session

##### 7.3.4.2.1 Vue d'ensemble de la façon dont le service pourrait être mis en œuvre dans un protocole

NOTE La description suivante présume un protocole tel que celui de la CEI 61158-6-5 qui met en œuvre ces services. D'autres protocoles mettant en œuvre ces services sont possibles.

Ce service est utilisé pour ouvrir une session Client/Serveur entre une session de point d'extrémité client et une de point d'extrémité serveur. La paire d'adresses réseau source et destination identifie chaque session du point d'extrémité chaque adresse étant composée de l'adresse réseau de l'appareil et d'un sélecteur interne d'appareil.

Les sessions des points d'extrémité de type client sont configurées avec le PD Tag du service en lieu et place de l'adresse réseau du serveur. Ils peuvent utiliser le service Find Tag Query de SM pour déterminer l'adresse réseau de destination ou ils peuvent l'obtenir à partir des annonces périodiques (Periodic Annunciation) envoyées par le serveur.

L'utilisation du service Find Tag Query peut également être nécessaire pour rouvrir une session après une défaillance. Dans ce cas, la cause de la défaillance peut être une défaillance du serveur. Si le serveur est redondant, le secondaire prendra la main, mais il utilisera sa propre adresse réseau. Par conséquent, le client a besoin de déterminer si l'adresse réseau associée au PD Tag du serveur a changé.

Une fois que l'adresse réseau du serveur a été déterminée, le client envoie le message de demande de service FDA Open Session au FDA Selector placé à l'adresse de l'appareil réseau.

Le champ Version dans l'en-tête de message FDA dans le message de demande indique la version du FDA qui est demandée pour la session. Le répondeur peut négocier de baisser la version.

Le champ Options dans l'en-tête de message dans le message de demande indique les options qui sont demandées pour la session. Le bit Instance Id bit est toujours mis. Le répondeur peut refuser toutes les autres options. Le fait de retourner 0 dans la position de bit représentant l'option indique le refus.

Si l'option numéro de message est demandée, le champ de fin contient la valeur initiale devant être utilisée par les deux points d'extrémité de la session. Cette valeur est retournée dans le message de réponse. Si le demandeur ne reçoit pas de réponse à la demande et choisit d'envoyer à nouveau la demande, il incrémente le numéro de message à la fin, tout en maintenant le même invoke id que l'original. De ce fait, la demande est livrée à la session du point d'extrémité créée si la première demande avait été reçue et une réponse positive avait été retournée. Par contre, si la demande initiale n'avait pas été reçue ou si elle avait donné lieu au retour d'une réponse négative (qui n'avait pas été reçue), la nouvelle demande est traitée comme une nouvelle demande, car il n'existe aucune session de point d'extrémité pour traiter le message.

L'adresse FDA dans l'en-tête de message n'est pas utilisée et elle est mise à 0.

L'échange réussi des messages de demande et de réponse de service Open Session donne lieu à l'établissement d'une session qui peut être utilisée pour envoyer des messages de demande et de réponse. Si les services sans connexion sont utilisés, le serveur retourne la réponse à partir d'un sélecteur sans connexion non utilisé précédemment. Ce port est alors utilisé pour envoyer et recevoir tous les messages ultérieurs pour la session.

Si le répondeur n'a pas un indice de session (Session Index) disponible pour la nouvelle session, une réponse négative est retournée avec la classe d'erreurs = “resource” et le code d'erreur = “max sessions exceeded”.

Si le répondeur n'a pas les ressources pour prendre en charge la nouvelle session, une réponse négative est retournée avec la classe d'erreurs = “resource” et le code d'erreur = “object creation failure” ou quelque autre code d'erreur approprié au sein de la classe “resource”.

Le message de demande de service Open Session achemine les attributs de session pour validation et utilisation par le répondeur. Si le répondeur est capable de supporter certain de ceux-ci, il est permis de les négocier, tels que définis dans les paramètres de message de demande dans le Tableau 121 ci-dessous. Si le demandeur n'est pas prêt à opérer avec les attributs négociés retournés par le récepteur, il est libre de ne pas ouvrir la session. Il peut alors réémettre une demande de service Open Session avec des attributs de session différents ou, s'il utilise des services orientés connexion, il peut fermer la connexion associée pour fermer la session, selon le cas

Si une demande est reçue avec des valeurs de paramètre non valides ou non prises en charge, une réponse négative est retournée avec une classe d'erreurs "service" et un code d'erreur "parameter-inconsistent" pour le paramètre incriminé. Cette réponse négative utilise également la valeur de code supplémentaire égale à 1 et la description supplémentaire pour proposer des valeurs acceptables pour chacun des paramètres indiqués ci-dessous qui peuvent être utilisés pour ouvrir la session. Pour acheminer ces valeurs, les 16 octets de la description supplémentaire sont interprétés comme quatre nombres entiers Unsigned32 au lieu d'un VisibleString. L'emplacement de chaque valeur de paramètre dans les 16 octets est spécifié dans les descriptions ci-dessous.

Si le répondeur reçoit un message de demande de service Open Session pour une session sur laquelle il a déjà envoyé un message de réponse de service Open Session, il ferme la session.

Si le répondeur n'a pas un indice de session (Session Index) disponible pour la nouvelle session, une réponse négative est retournée avec la classe d'erreurs = “resource” et le code d'erreur = “max sessions exceeded”.

Si le répondeur n'a pas les ressources pour prendre en charge la nouvelle session, une réponse négative est retournée avec la classe d'erreurs = “resource” et le code d'erreur = “object creation failure” ou quelque autre code d'erreur approprié au sein de la classe “resource”.

Les sessions ouvertes pour NMA Configuration Use ('utilisation de la configuration NMA) sont appelées "sessions de configuration". Une seule session de configuration est autorisée à la fois. Si une demande est reçue pour ouvrir une session de configuration et une est déjà ouverte, ou, en option, si une VCR vers une MIB de Type 5 ou de Type 9 avec des services de mise à jour pris en charge est déjà ouverte, la session de configuration est refusée. L'agent FDA retourne une réponse négative avec la classe d'erreurs = "access" et le code d'erreur = "config-access-already-open".

Un soin particulier est nécessaire pour négocier la période d'inactivité pour les sessions de configuration qui utilisent des services sans connexion. Si le client a une défaillance, il est interdit au serveur de laisser un autre client de configuration établir une session tant que la durée d'inactivité négociée n'a pas expiré.

L'ouverture de sessions de type éditeur/abonné et de type Report Distribution est locale et n'entraîne pas l'échange de messages FDA. Par conséquent, les paramètres de message Open Session pour ces types de session sont configurés dans la NMIB au lieu d'être négociés.

#### 7.3.4.2.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 121.

**Tableau 121 – Service Open session**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| Source Address | | M | | |
| Destination Address | M | M (=) | | |
| Session Index | M | M (=) | | |
| Max Buffer Size | M | M (=) | | |
| Max Message Length | M | M (=) | | |
| NMA Configuration Use | M | M (=) | | |
| Inactivity Close Time | M | M (=) | | |
| Transmit Delay Time | M | M (=) | | |
| PD Tag | M | M (=) | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Session Index | | | M | M (=) |
| Max Buffer Size | | | M | M (=) |
| Max Message Length | | | M | M (=) |
| NMA Configuration Use | | | M | M (=) |
| Inactivity Close Time | | | M | M (=) |
| Transmit Delay Time | | | M | M (=) |
| PD Tag | | | M | M (=) |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| Source Address | | | | M |
| Destination Address | | | M | M (=) |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la demande a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la demande avait été envoyée.

**Session Index**

L'indice de session (Session Index) est l'indice OD de la description des sessions des points d'extrémité de la session du point d'extrémité client, s'il en existe une. Sinon, 0 est utilisé.

**Max Buffer Size**

Ce paramètre définit la longueur maximale du buffer en octets (le buffer peut contenir des messages FDA concaténés) que l'expéditeur de ce message peut envoyer ou recevoir sur cette session. Ce paramètre peut être négocié à la baisse (mais pas à la hausse) par le répondeur en utilisant le paramètre Max Buffer Size. Si la négociation échoue pour un paramètre quelconque, une valeur acceptable pour ce paramètre est retournée dans le champ de description supplémentaire comme un Unsigned32 au décalage 0. Si la valeur demandée est prise en charge, mais un autre paramètre a échoué à la négociation, la valeur demandée est retournée au décalage 0.

**Max Message Length**

Ce paramètre définit la longueur maximale en octets des messages devant être envoyés sur cette session par l'expéditeur de ce message. Il est utilisé par la session du point d'extrémité destinataire comme son attribut Max Message Length. Si la négociation échoue pour un paramètre quelconque, une valeur acceptable pour ce paramètre est retournée dans le champ de description supplémentaire comme un Unsigned32 au décalage 4. Si la valeur demandée est prise en charge, mais un autre paramètre a échoué à la négociation, la valeur demandée est retournée au décalage 4.

**NMA Configuration Use**

0 = NMA Configuration Not Permitted (Configuration NMA interdite)

1 = NMA Configuration Permitted (Configuration NMA permise)

Ce paramètre ne peut pas être négocié.

**Inactivity Close Time**

Ce paramètre identifie la durée, en secondes, pendant laquelle la session reste ouverte sans recevoir de message. Après avoir connu une inactivité sur la session pendant cette durée, la session du point d'extrémité est fermée, ainsi que toute l'activité VCR associée à la session du point d'extrémité. Le répondeur peut négocier cette valeur à la baisse. La valeur 0 est interdite. Si la négociation échoue pour un paramètre quelconque, une valeur acceptable pour ce paramètre est retournée dans le champ de description supplémentaire comme un Unsigned32 au décalage 8. Si la valeur demandée est prise en charge, mais un autre paramètre a échoué à la négociation, la valeur demandée est retournée au décalage 8.

**Transmit Delay Time**

Ce paramètre est utilisé pour établir l'attribut Transmit Delay Time de la session du point d'extrémité de réception. Sa valeur ne peut pas être négociée. Sa valeur est exprimée en millisecondes. Si la négociation échoue pour un paramètre quelconque, une valeur acceptable pour ce paramètre est retournée dans le champ de description supplémentaire comme un Unsigned32 au décalage 12. Si la valeur demandée est prise en charge, mais un autre paramètre a échoué à la négociation, la valeur demandée est retournée au décalage 12.

**PD Tag**

PD Tag du serveur. Si le PD Tag dans le message de demande ne concorde pas avec le PD Tag du serveur, le serveur rejette la demande avec la classe d'erreurs “access” et le code d'erreur “object-access-denied”.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

**Session Index**

L'indice de session (Session Index) est l'indice d'OD de la session nouvellement ouverte.

**Max Buffer Size**

Ce paramètre définit la longueur négociée du buffer en octets (le buffer peut contenir des messages de FDA concaténés) que l'expéditeur de ce message peut envoyer ou recevoir sur cette session.

**Max Message Length**

Ce paramètre définit la longueur maximale en octets des messages devant être envoyés sur cette session par l'expéditeur de ce message. Il est utilisé par la session du point d'extrémité destinataire comme son attribut Max Message Length.

**NMA Configuration Use**
0 = NMA Configuration Not Permitted (Configuration NMA interdite)

1 = NMA Configuration Permitted (Configuration NMA permise)

**Inactivity Close Time**

Ce paramètre identifie la durée, en secondes, pendant laquelle la session reste ouverte sans recevoir de message. Après avoir connu une inactivité sur la session pendant cette durée, la session du point d'extrémité est fermée, ainsi que toute l'activité VCR associée à la session du point d'extrémité. Le répondeur peut négocier cette valeur à la baisse. La valeur 0 est interdite.

**Transmit Delay Time**

Ce paramètre est utilisé pour établir l'attribut Transmit Delay Time dans le point d'extrémité de session de réception. Sa valeur est exprimée en millisecondes.

**PD Tag**

PD Tag du serveur.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**Source Address**

Ce paramètre est l'adresse à partir de laquelle la réponse a été envoyée.

**Destination Address**

Ce paramètre est l'adresse vers laquelle la réponse est envoyée.

#### 7.3.4.3 Service Idle

##### 7.3.4.3.1 Vue d'ensemble du service

Ce service est utilisé par des points d'extrémité VCR client/serveur pour entretenir la VCR. La réception d'un message Idle informe le destinataire que l'expéditeur est présent. Il peut être envoyé au moyen de services sans connexion ou au moyen de services orientés connexion. L'adresse de FDA contient l'indice d'OD du point d'extrémité VCR serveur établi dynamiquement. Si ce point d'extrémité n'existe pas, un message d'erreur est retourné avec la classe d'erreurs = "access" et le code d'erreur = "unrecognized FDA Address".

##### 7.3.4.3.2 Primitives du service

Les paramètres de service pour ce service sont montrés dans le Tableau 122.

**Tableau 122 – Service Idle session**

| Nom de paramètre | Req | Ind | Rsp | Cnf |
|---|---|---|---|---|
| Argument | | | | |
| Invoke ID | U | | | |
| VCR Id | | M | | |
| | | | | |
| Result (+) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| VCR Id | | | | M |
| | | | | |
| Result (-) | | | S | S (=) |
| Invoke ID | | | | U (=) |
| VCR Id | | | | M |
| Error Info | | | 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.

**Argument**

L'argument contient les paramètres de la demande du service.

**VCR Id**

Ce paramètre identifie la VCR du service.

**Result (+)**

Ce paramètre de type sélection indique que la demande de service a réussi.

**VCR Id**

Ce paramètre identifie la VCR du service.

**Result(–)**

Ce paramètre de type sélection indique que la demande de service a échoué.

**VCR Id**

Ce paramètre identifie la VCR du service.

### 7.4 Résumé des classes de Type 9 et de Type 5 de la FAL

Le Tableau 123 contient un résumé des classes de FAL définies. Les valeurs de Class ID sont définies dans les paragraphes correspondants de l'Article 15 (Type 9).

**Tableau 123 – Résumé des classes de FAL**

| FAL ASE | Classe | Class ID |
|---|---|---|
| VFD | VFD | — |
| | VCR | |
| Data type (Type de données) | Fixed Length & String Data type | Voir Type 9 |
| | Structure Data type | Voir Type 9 |
| Object Dictionary (Dictionnaire d'objets) | OD Description | — |
| | Object Dictionary | — |
| Type 5 AR (FDA Session) | Client/serveur | — |
| | Publisher/subscriber | — |
| | Report Distribution | — |
| Type 9 Variable | Simple Variable | Voir Type 9 |
| | Array Variable | Voir Type 9 |
| | Record Variable | Voir Type 9 |
| | Variable List | Voir Type 9 |
| Type 9 Event | Event | Voir Type 9 |
| Type 9 Load Region | Load Region | Voir Type 9 |
| Type 9 Function Invocation | Function Invocation | Voir Type 9 |
| SMK | SMK | — |
| LAN Redundancy | LAN Redundancy | — |

### 7.5 Services autorisés de Type 9 et de Type 5 de la FAL par chaque rôle AREP

Le Tableau 124 définit les combinaisons valides de services et de rôles d'AREP (quelles APDU de service et quels AREP avec le rôle spécifié peuvent envoyer ou recevoir). Les colonnes "Unc" et "Cnf" indiquent si le service figurant dans la colonne de gauche est non confirmé (unconfirmed, Unc) ou confirmé (confirmed, Cnf).

**Tableau 124 – Services par rôle d'AREP**

| | Unconfirmed | Confirmed | Client | | Server | | Publish | Subscribe | |
|---|---|---|---|---|---|---|---|---|---|
| **Services de FAL** | | | **req** | **rcv** | **req** | **rcv** | **req rcv** | **req** | **rcv** |
| **ASE Type 9 Mgt** | | | | | | | | | |
| Create | | X | X | X | X | X | | | |
| Delete | | X | X | X | X | X | | | |
| Get Attributes | | X | X | X | X | X | | | |
| Get Attribute List | | X | X | X | X | X | | | |
| Set Attributes | | X | X | X | X | X | | | |
| Begin Set Attributes | | X | X | X | X | X | | | |
| End Set Attributes | | X | X | X | X | X | | | |
| **ASE VFD** | | | | | | | | | |
| Identify | | X | X | X | X | X | | | |
| Get Status | | X | X | X | X | X | | | |
| Status Notification | X | | X | X | X | X | X | | X |
| Initiate | | X | X | X | X | X | | | |
| Terminate | | X | X | X | X | X | | | |
| **ASE FDA Session** | | | | | | | | | |
| Confirmed Send | | | X | X | X | X | | | |
| Unconfirmed Send | | | X | X | X | X | X | | X |
| Open Session | | | X | | | X | | | |
| **ASE Type 9 AR** | | | | | | | | | |
| AR-Confirmed Send | | | X | X | X | X | | | |
| AR-Unconfirmed Send | | | X | X | X | X | X | | X |
| AR-Associate | | | X | | | X | | | |
| AR-Abort | | | X | X | X | X | X | X | X |
| **ASE Type 9 Variable** | | | | | | | | | |
| Read | | X | X | X | X | X | | | |
| Write | | X | X | X | X | X | | | |
| Information Report | X | | X | X | X | X | X | | X |
| **ASE Type 9 Event** | | | | | | | | | |
| Confirmed Ack Event | | X | X | X | X | X | | | |
| Enable Event | | X | X | X | X | X | | | |
| Event Notification | X | | X | X | X | X | X | | X |
| **ASE Type 9 Domain** | | | | | | | | | |
| Initiate Load | | X | X | X | X | X | | | |
| Push Segment | | X | X | X | X | X | | | |
| Pull Segment | | X | X | X | X | X | | | |
| Terminate Load | | X | X | X | X | X | | | |

| Services de FAL | Unconfirmed | Confirmed | Client | | Server | | Publish | Subscribe |
|---|---|---|---|---|---|---|---|---|
| | | | req | rcv | req | rcv | req rcv | req rcv |
| **ASE Type 9 Program Invocation** | | | | | | | | |
| Start | | X | X | X | X | X | | |
| Stop | | X | X | X | X | X | | |
| Resume | | X | X | X | X | X | | |
| Reset | | X | X | X | X | X | | |
| Kill | | | X | X | X | X | | |
| **ASE LAN Redundancy** | | | | | | | | |
| Diagnostic Message | X | | | | | | X | X |
| **ASE SMK** | | | | | | | | |
| Find Tag Query | X | | X | X | X | X | X | X |
| Find Tag Reply | X | | X | X | X | X | X | X |
| Identify | | X | X | X | X | X | | |
| Annunciate | X | | | | | | X | X |
| Device Assignment | | X | X | X | X | X | X | X |
| Clear Address | | X | X | X | X | X | X | X |

# Bibliographie

CEI 61784-1, *Réseaux de communication industriels – Profils – Partie 1: Profils de bus de terrain*

CEI 61784-2, *Réseaux de communication industriels – Profils – Partie 2: Profils de bus de terrain supplémentaires pour les réseaux en temps réel basés sur l'ISO/CEI 8802-3*

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

_____

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

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