IEC 61158-6-5

Edition 2.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

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

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-5: Spécification du protocole de la couche application – Éléments de type 5

IEC 61158-6-5:2014-08(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 It also gives information on projects, replaced and withdrawn publications.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 61158-6-5

Edition 2.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

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

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-5: Spécification du protocole de la couche application – Éléments de type 5

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XE

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1759-7

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-5: Application layer protocol specification – 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 itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

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

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

International Standard IEC 61158-6-5 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:

- Add support for message padding
- Clarified encoding rules
- Clarified open session service
- Time synchronization now present in annunciation message
- Additional redundancy options in annunciation message

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

FDIS	Report on voting
65C/764/FDIS	65C/774/RVD

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

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

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

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

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

INTRODUCTION

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

The application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

- as a guide for implementors and designers;
- for use in the testing and procurement of equipment;
- as part of an agreement for the admittance of systems into the open systems environment;
- as a refinement to the understanding of time-critical communications within OSI.

This standard is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this standard together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems may work together in any combination.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-5: Application layer protocol specification – 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 behavior provided by the Type 5 fieldbus Application Layer in terms of

a) the abstract syntax defining the application layer protocol data units conveyed between communicating application entities,
b) the transfer syntax defining the application layer protocol data units conveyed between communicating application entities,
c) the application context state machine defining the application service behavior visible between communicating application entities; and
d) the application relationship state machines defining the communication behavior visible between communicating application entities; and.

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

1) define the wire-representation of the service primitives defined in IEC 61158-5-5, and
2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 5 IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI Application Layer Structure (ISO/IEC 9545).

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

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

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

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-5.

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 protocols standardized in IEC 61158-6 series.

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. Conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

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

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

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

IEC 61158-5-5, *Industrial communication networks – Fieldbus specifications – Part 5-5: Application layer service definition – Type 5 elements*

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

ISO/IEC 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for 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*

IETF RFC 791, *Internet Protocol;* available at <http://www.ietf.org>

[1] Withdrawn

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 Terms and definitions from ISO/IEC 7498-1

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

3.1.2 Terms and definitions from ISO/IEC 9545

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

3.1.3 Terms and definitions from ISO/IEC 8824

a) object identifier
b) type
c) value
d) simple type
e) structured type
f) component type
g) tag
h) Boolean type
i) true
j) false
k) integer type
l) bitstring type
m) octetstring type
n) null type
o) sequence type
p) sequence of type
q) choice type
r) tagged type
s) any type
t) module
u) production

3.1.4 Terms and definitions from ISO/IEC 8825

a) encoding (of a data value)
b) data value
c) identifier octets (the singular form is used in this standard)
d) length octet(s) (both singular and plural forms are used in this standard)
e) contents octets

3.2 IEC 61158-1 terms

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

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

3.2.2
application layer interoperability
capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.2.3
application object
object class that manages and provides the run time exchange of messages across the network and within the network device

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

3.2.4
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.2.5
application process identifier
component that distinguishes multiple application processes used in a device

3.2.6
application process object
component of an application process that is identifiable and accessible through an FAL application relationship

Note 1 to entry: Application process object definitions are composed of a set of values for the attributes of their class (see the definition for Application Process Object Class Definition). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions, and to dynamically create and delete application objects and their corresponding definitions.

3.2.7
application process object class
class of application process objects defined in terms of the set of their network-accessible attributes and services

3.2.8
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

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

3.2.9
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.2.10
application relationship endpoint
context and behavior of an application relationship as seen and maintained by one of the application processes involved in the application relationship

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

3.2.11
attribute
description of an externally visible characteristic or feature of an object

Note 1 to entry: The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behaviour of an object. Attributes are divided into class attributes and instance attributes.

3.2.12
behaviour
indication of how the object responds to particular events

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

3.2.13
class
set of objects, all of which represent the same kind of system component

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

3.2.14
class attributes
attribute that is shared by all objects within the same class

3.2.15
class code
unique identifier assigned to each object class

3.2.16
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service.

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

3.2.17
client
(a) object which uses the services of another (server) object to perform a task

(b) initiator of a message to which a server reacts, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

3.2.18
conveyance path
unidirectional flow of APDUs across an application relationship

3.2.19
cyclic
term used to describe events which repeat in a regular and repetitive manner

3.2.20
dedicated AR
AR used directly by the FAL User

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

3.2.21
device
physical hardware connection to the link

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

3.2.22
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.2.23
dynamic AR
AR that requires the use of the AR establishment procedures to place it into an established state

3.2.24
endpoint
one of the communicating entities involved in a connection

3.2.25
error
a discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.2.26
error class
general grouping for error definitions

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

3.2.27
error code
identification of a specific type of error within an error class

3.2.28
FAL subnet
networks composed of one or more data link segments

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

3.2.29
logical device
certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.2.30
management information
network-accessible information that supports managing the operation of the fieldbus system, including the application layer

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

3.2.31
network
series of nodes connected by some type of communication medium

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

3.2.32
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.2.33
pre-defined AR endpoint
AR endpoint that is defined locally within a device without use of the create service

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

3.2.34
pre-established AR endpoint
AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.2.35
publisher
role of an AR endpoint in which it transmits APDUs onto the fieldbus for consumption by one or more subscribers

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

3.2.36
server
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

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

3.2.37
service
operation or function than an object and/or object class performs upon request from another object and/or object class

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

3.2.38
subscriber
role of an AREP in which it receives APDUs produced by a publisher

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

3.3 Abbreviations and symbols

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process

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

3.4 Conventions

3.4.1 General concept

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 three parts: its class definitions, its services, and its protocol specification. The first two are contained in IEC 61158-5 series. The protocol specification for each of the ASEs is defined in this standard.

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in IEC 61158-5 standard. The service specification defines the services that are provided by the ASE.

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

3.4.2 Conventions for class definitions

The data-link layer mapping definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is defined in IEC 61158-5-5.

3.4.3 Abstract syntax conventions

When the "optionalParametersMap" parameter is used, a bit number which corresponds to each OPTIONAL or DEFAULT production is given as a comment.

3.5 Conventions used in state machines

The state machines are described in Table 1.

Table 1 – Conventions used for state machines

#	Current state	Event / condition => action	Next state
Name of this transition.	The current state to which this state transition applies	Events or conditions that trigger this state transaction. => The actions that are taken when the above events or conditions are met. The actions are always indented below events or conditions	The next state after the actions in this transition is taken

The conventions used in the state machines are as follows:

:= Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.

xxx A parameter name.

Example:
Identifier := reason
1) means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'

"xxx" Indicates fixed value.

Example:
Identifier := "abc"
2) means value "abc" is assigned to a parameter named 'Identifier.'
3)

= A logical condition to indicate an item on the left is equal to an item on the right.

< A logical condition to indicate an item on the left is less than the item on the right.

> A logical condition to indicate an item on the left is greater than the item on the right.

<> A logical condition to indicate an item on the left is not equal to an item on the right.

&& Logical "AND"

|| Logical "OR"

This construct allows the execution of a sequence of actions in a loop within one transition. The loop is executed for all values from start_value to end_value.

Example:
for (Identifier := start_value to end_value)
actions
4) endfor

This construct allows the execution of alternative actions depending on some condition (which might be the value of some identifier or the outcome of a previous action) within one transition.

Example:
If (condition)
actions
else
actions
5) endif

Readers are strongly recommended to refer to the subclauses for the AREP attribute definitions, the local functions, and the FAL-PDU definitions to understand protocol machines. It is assumed that readers have sufficient knowledge of these definitions, and they are used without further explanations.

4 Protocol

4.1 Overview

The Type 5 services and protocols are provided by the Type 5 AE, referred to as the *Field Device Access (FDA) Agent* throughout this specification.

The FDA Agent conveys Type 9 APDUs using Type 9 Application Relationships, referred to as *FDA Sessions*, or more simply as *sessions* throughout this Type 5 specification.

The FDA Agent may provide gateway functions that map between Type 5 services and corresponding services that operate over a Type “X” DLL. Networks that employ the Type “X” DLL are referred to as *Type “X”* networks.

4.2 FAL syntax description

4.2.1 PDU abstract syntax

The abstract syntax of APDUs is combined with their transfer syntax and is specified in 4.3.

4.2.2 Abstract syntax of data types

The abstract syntax of data types is combined with their transfer syntax and is specified in 4.3.

4.3 Transfer syntax

4.3.1 General

The transfer syntax combines the specification of the abstract syntax and their encodings as a set of fixed format APDUs. Each APDU contains a header and an APDU body. An optional trailer is also defined. The presence of the trailer is indicated in the APDU header.

The format of the APDU header and trailer is specified in 4.3.3 and 4.3.4. The APDU header contains the fields common to all messages that are mandatory. The APDU trailer contains the fields common to all APDUs that are optional.

The formats of the APDU bodies are specified in 4.3.5. If a service request or response APDU has no parameters, it is stated in the subclause that describes the APDU body. If the service has no response or error APDU body associated with it, then the corresponding subclause is not present. Common error parameters are used in many service-specific APDUs. Their definitions are specified in 4.3.6.1.

User data, when present in an APDU, is defined as a service-specific APDU body parameter. The content and length of the user data are dependent upon the service and the object being accessed. Therefore, neither is specified as part of the APDU body. However, the user data length can be calculated from the APDU length contained in the APDU header.

The APDU header, trailer, and bodies are composed of a subset of data types defined in IEC 61158-5. The encodings for these data types are defined next, followed by the definitions of the header, trailer, and APDU bodies.

4.3.2 Data type encodings

The following data types are used in APDUs as listed in Table 2. Other data types may be conveyed as user data.

Table 2 – Data types

Data Type	Octet Length	Comment
Boolean	1	0 = False, Non-Zero = TRUE
Integer8	1	
Integer16	2	
Unsigned8	1	
Unsigned16	2	
Unsigned32	4	
VisibleString	-	The length of a VisibleString is defined by the field definition.
OctetString	-	The length of an OctetString is defined by the field definition.
Time value	8	

For all data types, the bit and octet order of transmission is as defined in Type 9. The unsigned integer value for each bit position is shown in Table 3.

NOTE The resulting encoding is the same as that the encoding specified in Appendix B of RFC 791 – Internet Protocol. The bit and byte order specified in RFC 791 Appendix B is the big endian octet ordering (most significant byte first for integers) and bit ordering within bytes is most significant bit first. Although the bit numbering is opposite of that specified in FMS, both specify the high order bit (the leftmost bit) as the most significant bit. In the descriptions that follow, bit 1 is the most significant bit in each byte. The unsigned integer value for each bit position is shown in the table below.

Table 3 – Data types

Bit Position	Hex Value	Decimal Value
8	0x80	128
7	0x40	64
6	0x20	32
5	0x10	16
4	0x08	8
3	0x04	4
2	0x02	2
1	0x01	1

4.3.3 APDU header

4.3.3.1 APDU header format

The header is the same for each APDU and has a fixed length.

All header fields are mandatory.

The format of the header is shown in Table 4.

Table 4 – APDU header format

Parameter name	Octet offset	Data type	Octet length	Description
Type 5 APDU Version	0	Unsigned8	1	Specifies the Version number of the Type 5 APDU format. The version number of the Type 5 APDUs specified in this document is 1. 1 = Version 1
Options	1	Unsigned8	1	Bit 8: 1 = APDU Number present in the Trailer Bit 7: 1 = Invoke Id present in the Trailer (this bit shall be set for all client / server sessions) Bit 6: 1 = Time Stamp present in the Trailer Bit 5: Reserved, set to 0 Bit 4: 1 = Extended Control Field present in the Trailer Bit 3-1: Pad Length = Specifies the number of pad bytes that are inserted between the user data and the trailer fields. Each pad byte is set to binary zero.
ASE Id And Confirmed Msg Type	2	Unsigned8	1	Bits 8–3: ASE Id – Used to specify the ASE whose service is being conveyed by the APDU. They are interpreted as a 6-bit unsigned integer with the following values: 0 = Unused 1 = AR ASE 2 = SMK ASE 3 = Type 9 ASEs/Type 5 VFD ASE 4 = LAN Redundancy ASE 5-63 : Reserved Bits 1, 2: Confirmed Msg Type – Used to differentiate between request, response, and error APDUs. Not used with unconfirmed services. These bits are interpreted as a 2-bit unsigned integer with the following values: 0 = Request APDU 1 = Response APDU 2 = Error APDU 3 = Reserved
Service	3	Unsigned8	1	Specifies the service within the specified protocol. Bit 8: Confirmed Flag (Confirmed=1) and (Unconfirmed=0). When this flag is set for a request APDU, the corresponding response (either a response APDU or an error APDU) is always returned after the request has been processed to indicate the result of the processing. Bits 7-1 Service Id of the service (7-bit unsigned integer). The value used for each Service Id is specified in its APDU Body specification below. It is the number in parenthesis that follows the service name in the header for each service. All other values are reserved.
FDA Address	4	Unsigned32	4	The format and use of the FDA Address is dependent on the ASE Id and the type of S-VFD Context. The next subclause describes this field.
APDU Length	8	Unsigned32	4	Specifies the number of octets contained in the entire APDU, including the header and the trailer.

4.3.3.2 FDA address use

4.3.3.2.1 FDA address use summary

Subclause 4.3.3.2.1 specifies the values for the FDA Address field of the header. See Table 5 for details. In the descriptions that follow, a Type 9 device is a device with the Type 9 AL and

a DLL whose addressing can be represented as a Type 1 DLL address. A Type 9 SMK is the SMK in a Type 9 device.

Table 5 – FDA address use

ASE	VCR	Bits	Use
AR	N/A	32-1	Service-specific.
LAN Redundancy	N/A	32-1	Not Used, set to 0.
SMK	N/A	32-17	Link Id Set to 0 if the destination SMK is located in the Type 5 device receiving the APDU. If the device is a Linking Device, and the destination SMK is in a Type 9 device attached to the Linking Device, then the FDA Address field contains the link id used to access the Type 9 device.
		16-1	Node.Selector The node value is the high order 8 bits and the selector value is the low order 8 bits, when taken as a16-bit unsigned integer value. When used to identify a Type 5 SMK (when the Link Id is zero), the node value is set to 0 and the selector value is set to 2. When used to identify a Type 9 SMK (when the Link Id is non-zero), this is the individual or group Type 9 node.selector address. See the description of the Find Tag Query APDU for a detailed description of its use of the FDA Address.
Type 9	Client/Server	32-17	Link Id If the server S-VFD is located in the Type 5 device, then this is 0. If the device is a Linking Device, and the destination VFD is in a Type 9 device attached to the Linking Device, then this is the Link Id of the link to which the Type 9 device is attached.
		16-1	Type 5 Selector Initiate Request: The use of this field is dependent on the Connect Option selected for the request. See the description of the Initiate APDU for the values used. Initiate Response: The Type 5 FAL AE returns the VCR Id of the initiated VCR. All Others: The Selector returned in the Initiate Response.
Type 9	Publisher / Subscriber	32-17	Link Id If the publisher VFD is located in the Type 5 device sending the APDU, then this is combined with the Selector bits to form a 32-bit flat Dlcep address. If the device is a Linking Device, and the publisher is in a Type 9 device attached to the Linking Device, then this is the Link Id of the link to which the publishing device is attached.
		16-1	Dlcep Selector If the publisher is located in the Type 5 device sending the APDU, then this is part of the 32-bit flat Dlcep address assigned to the published data. If the device is a Linking Device, and the publisher is in a Type 9 device attached to the Linking Device, then this is the Dlcep Selector of the publisher buffer.
Type 9	Report Distribution	32-17	Link Id If the report source VFD is located in the Type 5 device sending the APDU, then this is combined with the Selector bits to form a 32-bit flat DLSAP address. If the device is a Linking Device, and the report source is in a Type 9 device attached to the Linking Device, then this is the Link Id of the link to which the reporting device is attached.
		16-1	S-VFD Context Id or Dlsap Selector If the report source VFD is located in the Type 5 device sending the APDU, then this is part of the 32-bit flat Dlsap address assigned to the Report

ASE	VCR	Bits	Use
			Source VFD. If the device is a Linking Device, and the report source is in a Type 9 device attached to the Linking Device, then this is the Dlsap Selector of the report source.
NOTE The Link Id, Node, DLCEP Selector, and DLSAP Selectors are all Type 1 DLL address components.			

4.3.3.2.2 APDU sent by a client VCR endpoint

Table 6 illustrates the use of the FDA Address header field in APDUs sent by a client VCR endpoint.

Table 6 – FDA address header field APDUs sent by a client VCR endpoint

Message type	Destination AP	FDA address	
		Bits	Value
Type 5 Initiate Request APDU Connect Option 1	Non-MIB VFD of Type 5 device	Bits 32-17 Bits 16-1	0 (Local Type 5 Device) Generic Selector, Returned by Find Tag Query, or obtained by some other means
	Non-MIB VFD of Type 9 device connected to Linking Device	Bits 32-17 Bits 16-1	Link Id of Type 9 Link Client Type 9 VCR Index
Type 5 Initiate Request APDU Connect Option 2	MIB VFD of Type 5 device	Bits 32-17 Bits 16-1	0 (Local Type 5 Device) 0.0
	MIB VFD of Linking Device Type 9 Link Interface	Bits 32-17 Bits 16-1	Link Id of Type 9 Link 0.0
	MIB VFD of Type 9 device connected to Linking Device	Bits 32-17 Bits 16-1	Link Id of Type 9 Link Node.0
All Other Request Messages	MIB VFD of Type 5 device	Bits 32-1	FDA Address returned in Initiate Response message
	Non-MIB VFD	Bits 32-1	FDA Address returned in Initiate Response message
	MIB VFD of Linking Device Type 9 Link Interface	Bits 32-1	FDA Address returned in Initiate Response message
	MIB VFD of Type 9 device connected to Linking Device	Bits 32-1	FDA Address returned in Initiate Response message
	Non-MIB AP of Type 9 device connected to Linking Device	Bits 32-1	FDA Address returned in Initiate Response message

4.3.3.2.3 APDUs sent by a server AR VCR endpoint

Table 7 illustrates the use of the FDA Address header field in APDUs sent by a server VCR endpoint.

Table 7 – FDA address header field APDUs sent by a server VCR endpoint

Message type	Destination AP	FDA address	
		Bits	Value
Type 5 Initiate Response APDU Connect Option 1	Non-MIB VFD of Type 5 device	Bits 32-17 Bits 16-1	0 (Local Type 5 Device) Type 5 VCR Id[1]
	Non-MIB VFD of Type 9 device connected to Linking Device	Bits 32-17 Bits 16-1	Link Id of Type 9 Link Type 5 VCR Id[1]
Type 5 Initiate Response APDU Connect Option 2	MIB VFD of Type 5 device	Bits 32-17 Bits 16-1	0 (Local Type 5 Device) Type 5 VCR Id[1]
	MIB VFD of Linking Device Type 9 Link Interface	Bits 32-17 Bits 16-1	Link Id of Type 9 Link Interface Type 5 VCR Id[1]
	MIB VFD of Type 9 device connected to Linking Device	Bits 32-17 Bits 16-1	Link Id of Type 9 Link Type 5 VCR Id[1]
All Other Response Messages	MIB VFD of Type 5 device	Bits 32-1	Same as Request Message Same as Request Message
	Non MIB VFD of Type 5 device	Bits 32-1	Same as Request Message Same as Request Message
	MIB VFD of Linking Device Type 9 Link	Bits 32-1	Same as Request Message Same as Request Message
	MIB VFD of Type 9 device connected to Linking Device	Bits 32-1	Same as Request Message Same as Request Message
	Non-MIB VFD of Type 9 device connected to Linking Device	Bits 32-1	Same as Request Message Same as Request Message

NOTE The selector value (bits 17-32) returned is the Type 5 VCR Index of the dynamically created VCR.

4.3.3.2.4 APDUs sent by a publisher VCR endpoint

Table 8 illustrates the use of the FDA Address in messages sent by a publisher VCR endpoint.

Table 8 – FDA address header field APDUs sent by a publisher VCR endpoint

Message type	Publisher AP	FDA address	
		Bits	Value
Type 9 Request Message	AP in the Type 5 device	Bits 32-1	Flat 32-bit Type 9 address assigned to the published data buffer
	VFD of Type 9 device connected through a Type 9 link of a Linking Device	Bits 32-17 Bits 16-1	Link Id of Type 9 Link Node.Selector Dlcep of the published data buffer

4.3.3.2.5 Messages sent by a report source VCR endpoint

Table 9 illustrates the use of the FDA Address in APDUs sent by a report source VCR endpoint.

Table 9 – FDA address header field APDUs sent by a report source VCR endpoint

Message type	Report source AP	FDA address	
		Bits	Value
Type 9 Request Message	AP in the Type 5 device	Bits 32-1	Flat 32-bit Type 9 address assigned to the report source VFD.
	VFD of Type 9 device connected through a Type 9 link of a Linking Device	Bits 32-17 Bits 16-1	Link Id of Type 9 Link Node.Selector Dlsap of the source of the report

4.3.4 Trailer

FDA Sessions define which trailer fields are defined for the Type 9 APDUs that they convey. The use of trailer fields by SMK ASE and LAN Redundancy APDUs is service specific. See the SMK ASE and LAN Redundancy APDU definitions for their use of trailer fields.

The presence of trailer fields is negotiated for Client / Server sessions and configured for Publisher / Subscriber and Report Distribution sessions. Their presence is indicated in the Message Header Options field of the each message. The fields that are present are positioned after the user data in the order shown in Table 10.

Table 10 – APDU trailer fields

Field name	Order	Data type	Octet length	Description
APDU Number	1	Unsigned32	4	Monotonically increasing by 1, 0 based, number of the APDU within a specific session.
Invoke Id	2	Unsigned32	4	Request/response identifier used to match responses with requests. It is required for all request / response exchanges. Invoke Ids are unique within the context of an Application Relationship.
Time Stamp	3	OctetString	8	System Time at which sender created APDU (in Time value format). It can be used by receivers to calculate the one-way transfer time from the sender.
Extended Control Field	4	Unsigned32	4	Reserved For Future Use.

4.3.5 APDU Body

4.3.5.1 General

Each service has a set of APDU bodies that are conveyed in a fixed format in each APDU body. The length of the APDU body formats may vary only if the last field of the APDU body repeats or its length is variable. Variable length fields are supported only as the last field of a APDU body (before the APDU Trailer) so that their offset from the beginning of the APDU body is constant.

The number in parenthesis that follows the name of the service in each subclause header below is the value that is used for the Service Id field in the Type 5 APDU Header.

4.3.5.2 FDA Session ASE Services

4.3.5.2.1 Open Session Service (confirmed service Id = 1)

4.3.5.2.1.1 Service overview

This service is used to open a Client/Server AR between a client AR endpoint and a server AR endpoint. The source and destination address pair identifies each AR endpoint.

The Version field in the Type 5 APDU Header in the request message indicates the version of the FDA Agent that is being requested for the AR. The responder may negotiate the version down.

The Options field in the Type 5 APDU Header in the request indicates the options that are being requested for the AR. The Invoke 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 AR 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 AR endpoint to process the message.

The FDA Address in the APDU Header is not used and is set to 0.

If the server 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 not, the following applies.

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 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 table 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 TCP, close the associated TCP connection to close the session, as appropriate.

The Options field in the Message Header in the request message indicates the options that are being requested for the session. The Invoke 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. This will cause the session to close. 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.

For the PadLength option, the requester specifies the alignment boundary for the message. This is done by setting the PadLength to the maximum number of padding bytes that can be inserted and by padding this message accordingly. If this value requests 4-or 8-byte padding and the receiver does not support padding, it may reject the request using “invalid options”, or it may accept the request and return “000” in this field.

000 = No padding

011 = pad to a 4-byte boundary. The maximum number of pad bytes that can be inserted is three. Three bytes of pad are inserted into this message.

111 = pad to 8 byte boundary. The maximum number of pad bytes that can be inserted is seven. Seven bytes of pad are inserted into this message.

The server receives requests at a generic endpoint address and creates a new endpoint at a previously unused address to process each request and to return the response. The newly created endpoint is used for all subsequent APDUs sent and received on the AR.

If a request is received with invalid parameter or unsupported values, a negative response is returned with an error class of "service" and an 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 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.

Successful exchange of Open Session request and response messages results in the establishment of a session that may be used to send FMS request and response messages. If UDP is used, the server returns the response from a previously unused UDP port. This port is then used to send and receive all subsequent messages for the session.

ARs opened for MIB VFD Configuration Use are referred to as configuration ARs. Only one configuration AR is allowed at a time. If a request is received to open a configuration AR and one is already open, or if there is an Type 9 VCR to any MIB VFD (Type 5 or Type 9) with update services supported already open, then the configuration AR is refused. The AR ASE returns a negative response with error class = "service" and error code = "config-access-already-open".

Opening Publisher / Subscriber and Report Distribution ARs is local, and does not result in the exchange of Open Session Service APDUs. As a result, the Open Session Request APDU parameters for these AR types are preconfigured instead of negotiated.

4.3.5.2.1.2 Request APDU parameters

Table 11 lists the request APDU parameters.

Table 11 – Request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
AR Index	0	Unsigned32	4	This parameter contains the numeric identifier of the client AR endpoint, if one. If not, 0 is used. For the response, this parameter contains the numeric identifier of the newly opened AR.
Max Buffer Size	4	Unsigned32	4	This parameter defines the maximum buffer length in octets (the buffer may contain concatenated Type 9 APDUs) that the sender of this APDU may send or receive on this AR. 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	8	Unsigned32	4	This parameter defines the maximum length in octets of APDUs to be sent on this AR by the sender of this APDU. It is used by the receiving AR endpoint as its Max Message Length attribute. This parameter cannot be negotiated. The server may reject the value supplied by the client if it cannot support it using error class = “access” and error code = “unsupported max APDU length”. 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.
Reserved	12	Unsigned8	1	Unused, set to 0.
Configuration Use	13	Unsigned8	1	0 = Configuration Not Permitted 1 = Configuration Permitted
Inactivity Close Time	14	Unsigned16	2	This parameter identifies how long, in seconds, that the AR stays open without receiving an APDU. After experiencing inactivity on the AR for this period of time, the AREP is closed, along with all VCR activity associated with the AREP. 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	16	Unsigned32	4	This parameter is used to set the Transmit Delay Time attribute in the receiving AREP. 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

Parameter name	Octet offset	Data type	Octet length	Description
				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.
SMK PD Tag	20	VisibleString	32	SMK PD Tag of the server. If the SMK 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”.

4.3.5.2.1.3 Response APDU parameters

Same as Request APDU

4.3.5.2.1.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.2.2 Idle (confirmed service Id = 3)

4.3.5.2.2.1 Service overview

This APDU is used to inform the receiver that the sender is present.

4.3.5.2.2.2 Request APDU parameters

None.

4.3.5.2.2.3 Response APDU parameters

None.

4.3.5.3 SMK ASE services

4.3.5.3.1 SM find tag query (unconfirmed service Id = 1)

4.3.5.3.1.1 Service overview

The network address used to send the FIND_TAG_QUERY message and the FDA Address in the Type APDU Header together indicate which SMKs are to receive and process the message.

The Find Tag Query message may be sent to one or more SMKs in Type 5 and/or Type 9 devices. When sent to a linking device, the linking device Type 5 SMK responds if the linking device contains the queried object. The Type 9 interface SMKs in the linking device are used only to forward the query to Type 9 links. They never respond to the queries.

Although this service is unconfirmed, it uses the Invoke Id in the Message Trailer to allow it to be matched to the returned SM Find Tag Reply message.

Table 12 describes the distribution that may be requested for this service. The FDA Addresses in the table are shown in hexidecimal.

Table 12 – SMK FDA address values

FDA address meaning	FDA address		Destination address	Distribution
	LLLL	NN.SS		
Type 5 Device SMK	00 00	00.02	SMK Multicast	All Type 5 SMKs
			Individual	Individual Type 5 SMK
Type 5 and Type 9 SMK Local Link Group Address	00 00	01.09[1]	SMK Multicast	All Type 5 SMKs and all Type 9 SMKs
			Individual	Individual LD SMK and all Type 9 SMKs accessible through the LD. This does not include the SMKs of the linking device's Type 9 interfaces
Type 9 SMK Local Link Group Address	LL LL	01.09[1]	SMK Multicast	All Type 9 SMKs on Type 9 link LL LL
			Individual	All Type 9 SMKs on Type 9 link LL LL accessible through the LD. This does not include the SMK of the linking device's Type 9 LL LL interface
Individual Type 9 Address	LL LL	NN.02	Individual	Individual Type 9 SMK accessible through the LD

NOTE Specified in IEC 61158-4-1as the Type 1 link specific address for "the SMAEs of all DLEs on the link".

4.3.5.3.1.2 Request APDU parameters

Table 13 lists the SMK FDA address values.

Table 13 – SMK FDA address values

Parameter name	Octet offset	Data type	Octet length	Description
Query Type	0	Unsigned8	1	Indicates the type of query: 0 = PD Tag query for primary or non-redundant device 1 = VFD tag query 2 = Function-Block tag query 3 = Element Id query 4 = PD Tag/VFD Reference query 5 = Device Index query 6 = PD Tag query for secondary or member of redundant set All other values are reserved.
Reserved	1	Octetstring	3	Reserved, set to 0
Element Id Or VFD Reference	4	Unsigned32	4	Service Parameter When not used, set to binary 0.
PD Tag Or Object Tag	8	VisibleString	32	Service Parameter When not used, set to binary 0.
VFD Tag	40	VisibleString	32	Service Parameter When not used, set to binary 0.

4.3.5.3.2 SM find tag reply (unconfirmed service Id = 2)

4.3.5.3.2.1 Service overview

This message is used to reply to the SM Find Tag request message.

Although this service is unconfirmed, it uses the Invoke Id in the Type 5 APDU Trailer to allow it to be matched to the corresponding SM Find Tag Query message.

4.3.5.3.2.2 Request APDU parameters

Table 14 lists the request APDU parameters.

Table 14 – Request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Query Type	0	Unsigned8	1	Indicates the type of query: 0 = PD Tag query for primary or non-redundant device 1 = VFD tag query 2 = Function-Block tag query 3 = Element Id query 4 = PD Tag/VFD Reference query 5 = Device Index query 6 = PD Tag query for secondary or member of redundant set All other values are reserved.
Type 9 Node Address	1	Unsigned8	1	Service Parameter Set to 0 if the Reply is for a Type 5 device
Type 9 Link Id	2	Unsigned16	2	Service Parameter Set to 0 if the Reply is for a Type 5 device.
VFD Reference	4	Unsigned32	4	Service Parameter
Queried Object Numeric Id	8	Unsigned32	4	Service Parameter
Queried Object Network Address	12	Octetstring	16	Service Parameter
Queried Object OD Version Number	28	Unsigned32	4	Service Parameter
Queried Object Device ID	32	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Queried Object PD Tag	64	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Duplicate Detection State	96	Unsigned8	1	Service Parameter Encoded as follows. Bits 8-3: Reserved, set to 0. Bit 2: 1 = Duplicate PD Tag Detected 0 = Duplicate PD Tag Not Detected Bit 1: 1 = Duplicate Device Index Detected 0 = Duplicate Device Index Not Detected
Reserved	97	Unsigned8	1	Reserved, set to 0.
Number in List of FDA Addresses	98	Unsigned16	2	This parameter indicates how many FDA Address Selectors are contained in the List of FDA Address Selectors parameter. This value of this attribute is 0 if the Query Type is “physical-device query”.

Parameter name	Octet offset	Data type	Octet length	Description
List of FDA Addresses	100	Unsigned16	2x number in list	Service Parameter Each Selector is an Unsigned32 that immediately follows its predecessor in the series (no spaces or padding between them).

4.3.5.3.3 SM Identify (confirmed service Id = 3)

4.3.5.3.3.1 Service overview

This service is used to request the identity of a single Type 5 or Type 9 device or of a group of Type 5 devices.

Table 15 specifies the network addresses and FDA Addresses used with the SM Identify request message.

Table 15 – SMK FDA address values for SM identify

Identification of	FDA address LLLL.NN.SS	Network address	Request message delivered to
Type 5 SMK device, including	00 00.00.02	SM Multicast	All Type 5 SMKs
Live List Version Number List[a]		Individual	Individual Type 5 SMK
Linking Device Node Version Number List for Specific Type 9 Interface	LL LL.00.02	SM Multicast	All Linking Device Type 5 SMKs – the one with the requested Link Id responds
	LL LL.00.02	Individual	Linking Device Type 5 SMK
Type 9 device SMK	LL LL.NN.02	Individual	Linking Device Type 5 SMK -> Type 9 SMK

[a] The Live List Version Number List is returned by Linking Device SMKs.

4.3.5.3.3.2 Request APDU parameters

None.

4.3.5.3.3.3 Response APDU parameters

Same as SM Device Annunciation Request APDU parameters.

4.3.5.3.3.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.3.4 SMK clear address (confirmed service Id = 12)

4.3.5.3.4.1 Service overview

This service is always confirmed and, therefore, always uses the Invoke Id in the Message Trailer.

Table 16 specifies the Network Address and FDA Address used with the SM Clear Address request and response APDUs.

Table 16 – SMK FDA address values for SMK set assignment info request APDUs

Use	FDA address LLLL.NN.SS	Network address	Request path
Type 5 Clear Address	00 00.00.02	Individual	Type 5 Requester -> Type 5 SMK
Type 9 Clear Address	LL LL.NN.02	Individual	Type 5 Requester -> LD FDA Agent -> Type 9 SMK

4.3.5.3.4.2 Request APDU parameters

Table 17 specifies the request APDU parameters.

Table 17 – SMK clear address request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Device Id	0	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Physical Device Tag	32	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
InterfaceToClear	64	Unsigned8	1	Service Parameter Encoded as follows: 0 = Interface A 1 = Interface B 255 = All interfaces
Reserved	65	Octetstring	3	Reserved, set to 0.

4.3.5.3.4.3 Response APDU parameters

None.

4.3.5.3.4.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.3.5 SMK set assignment info (confirmed service Id = 14)

4.3.5.3.5.1 Service overview

This service is always confirmed and, therefore, always uses the Invoke Id in the Message Trailer.

Table 18 specifies the Network Address and FDA Address used with the SM Set Assignment Info request and response APDUs.

Table 18 – SMK FDA address values for SMK set assignment info request APDUs

Use	FDA address LLLL.NN.SS	Network address	Request path
Type 5 Set Assignment Info	00 00.00.02	Individual	Type 5 Requester -> Type 5 SMK
Type 9 Set Assignment Info	LL LL.NN.02	Individual	Type 5 Requester -> LD FDA Agent -> Type 9 SMK

4.3.5.3.5.2 Request APDU parameters

Table 19 specifies the request APDU parameters.

Table 19 – SMK set assignment info request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Device Id	0	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Physical Device Tag	32	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Type 9 New Address	64	Unsigned8	1	Service Parameter The field contains the new Type 9 address for the Type 9 device identified in the FDA Address of the Type 5 APDU Header. The value of this field is 0 if the FDA Address does not identify a Type 9 device or this APDU is not being used to change the Type 9 device's address.
Device Redundancy State	65	Unsigned8	2	Service Parameter Unused and set to 0 if the FDA Address identifies a Type 9 device. The value 0 indicates that the device is not participating in device redundancy. If it is to participate in device redundancy, the value of bit pair 2 and 1 is non-zero. The value of bit pair 4 and 3 is non-zero whenever bits 2 & 1 indicate "External Switchover Control Device". When the bit pair 1 and 2 indicate "External Switchover Control Device", bits 3 and 4 must be set to “2” to indicate that the current primary is to become the secondary, and one of the secondaries is to become the primary. Bit 8 = Reserved, set to 0 Bit 7 = Reserved, set to 0 Bit 6 = Reserved, set to 0 Bit 5 = Reserved, set to 0 Bits 4 & 3 –External Switchover Control Device Redundancy Role 0 = Not used 1 = Primary 2 = Secondary Bits 2 & 1 – Assigned Device Redundancy Type 0 = Non-redundant 1 = External Switchover Control 2 = Internal Switchover Control
LAN Redundancy Socket Address	66	Unsigned16	2	Service Parameter Unused and set to 0 if the FDA Address identifies a Type 9 device. Set to the Reserved LAN Redundancy port if the FDA Address identifies Type 5 device.
Annunciation Repeat Time	68	Unsigned32	4	Service Parameter Unused and set to 0 if the FDA Address identifies a Type 9 device.
Device Index	72	Unsigned16	2	Service Parameter Encoded as follows.

Parameter name	Octet offset	Data type	Octet length	Description
				0 = Index not assigned 1 to Max Device Index = Device Index
Max Device Index	74	Unsigned16	2	Service Parameter Unused and set to 0 if the FDA Address identifies a Type 9 device.
Operational Network Address	76	Octetstring	16	Service Parameter Unused and set to 0 if the FDA Address identifies an Type “X” device.
Reserved	92	Octetstring	3	Reserved, set to 0.
Clear Duplicate Detection State	95	Unsigned8	2	Service Parameter Unused and set to 0 if the FDA Address identifies a Type 9 device. Encoded as follows: Bits 8-3: Reserved, set to 0. Bit 2: 1 = Do not clear Duplicate PD Tag Detected 0 = Clear Duplicate PD Tag Detected Bit 1: 1 = Do not clear Duplicate Device Index Detected 0 = Clear Duplicate Device Index Detected

4.3.5.3.5.3 Response APDU parameters

Table 20 specifies the response APDU parameters.

Table 20 – SMK set assignment info response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Reserved	0	Unsigned16	2	Reserved, set to 0.
Max Device Index	2	Unsigned16	2	Service Parameter Unused and set to 0 if the FDA Address identifies a Type 9 device.
Annunciation Repeat Time	4	Unsigned32	4	Service Parameter Unused and set to 0 if the FDA Address identifies a Type 9 device.

4.3.5.3.5.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.3.6 SMK clear assignment info (confirmed service Id = 15)

4.3.5.3.6.1 Service overview

This service is always confirmed and, therefore, always uses the Invoke Id in the Message Trailer.

Table 21 specifies the Network Address and FDA Address used with the SM Clear Assignment Info request and response APDUs.

Table 21 – SMK FDA address values for SMK device clear assignment Info APDUs

Use	FDA address LLLL.NN.SS	Network address	Request path
Type 5 Clear Assignment Info	00 00.00.02	Individual	Type 5 Requester -> Type 5 SMK
Type 9 Clear Assignment Info	LL LL.NN.02	Individual	Type 5 Requester -> LD FDA Agent -> Type 9 SMK

4.3.5.3.6.2 Request APDU parameters

Table 22 defines the request APDU parameters.

Table 22 – SMK clear assignment info request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Device Id	0	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Physical Device Tag	32	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.

4.3.5.3.6.3 Response APDU parameters

None.

4.3.5.3.6.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.3.7 SMK device annunciation (unconfirmed service Id = 16)

4.3.5.3.7.1 Service overview

The request APDU body format defined below is used for the SM Device Annunciation service and also for the SM Identify service.

This service is always unconfirmed and no APDU Trailer fields are used.

Table 23 specifies the Network Address and FDA Address used with the SMK Device Annunciation request APDUs.

Table 23 – SMK FDA address values for SMK device annunciation request APDUs

Use	FDA address LLLL.NN.SS	Network address	Request path
Device Annunciation	00 00.00.02	SM Multicast	Type 5 SMK -> Configuration Applications

4.3.5.3.7.2 Request APDU parameters

Table 24 defines the request APDU parameters.

Table 24 – SMK device annunciation request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
SMK State	0	Unsigned8	1	Service Parameter Encoded as follows. Bits 2-8: 0 = Reserved 1 = NO_TAG 2 = OPERATIONAL 3-127 = Reserved Bit 1: 0 = Not Synchronized with SNTP Time Server 1 = Synchronized with SNTP Time Server
Device Type	1	Unsigned8	1	Service Parameter Encoded as follows. Each bit identifies a different device type. It is the destination device type for request messages and the source device for response messages. A combination of bits 8-6 and 2,1 may be set for devices. Bit 5 is used only for Type “X” devices, and only in SM Identify responses. Non Devices participating in device assignment set the value of bits 8 through 3 of this parameter to 0. Bit 8 = Linking Device Bit 7 = I/O Gateway Bit 6 = Field Device Bit 5 = Type “X” Device Bit 4 = Reserved Bits 3-1 – Redundant Device Type Capability 0 = Non-redundant 1 = External Switchover Control 2 = Internal Switchover Control 3 = External and Internal Switchover Control 4 = Not used 5 = Internal Switchover Control and Non-redundant device 6 = External Switchover Control and Non-redundant device 7 = External and Internal Switchover Control and Non-redundant device
Device Redundancy State	2	Unsigned8	2	Service Parameter Encoded as follows. Encoded as follows. Bit 8 = Reserved, set to 0 Bit 7 = Reserved, set to 0 Bit 6 = Reserved, set to 0 Bit 5 = Reserved, set to 0 Bits 4 & 3 – Device Redundancy Role 0 = Non-redundant 1 = Primary 2 = Secondary Bits 2 & 1 – Assigned Redundant Device Type 0 = Non-redundant

Parameter name	Octet offset	Data type	Octet length	Description
				1 = External Switchover Control 2 = Internal Switchover Control
Duplicate Detection State	3	Unsigned8	2	Service Parameter Encoded as follows. Bits 8-3: Reserved, set to 0. Bit 2: 1 = Duplicate PD Tag Detected 0 = Duplicate PD Tag Not Detected Bit 1: 1 = Duplicate Device Index Detected 0 = Duplicate Device Index Not Detected
Device Index	4	Unsigned16	2	Service Parameter Encoded as follows. 0 = Index not assigned 1 to Max Device Index = Device Index
Max Device Index	6	Unsigned16	2	Service Parameter
Operational Network Address	8	Octetstring	16	Service Parameter
Device Id	24	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Physical Device Tag	56	VisibleString	32	Service Parameter Unused octets in this field are set to ASCII space characters.
Annunciation Repeat Time	92	Unsigned32	4	Service Parameter
LAN Redundancy Socket Address	96	Unsigned16	2	Service Parameter
Reserved	98	Unsigned16	2	Service Parameter
APDU Version Number	100	Unsigned32	4	Service Parameter
Device Version Number	100	Unsigned32	4	Service Parameter
Number of Entries in Version Number List	104	Unsigned32	4	This parameter indicates how many version numbers are contained in the Version Number List field below. Its value is 0 if the device is not a linking device or a gateway.
Version Number List	108	Unsigned32	4 x Number of Entries	Service Parameter This field is present only for linking devices and I/O gateways. This field contains a list of numbers. Each is an Unsigned32 number. The number of entries in the list is specified by the field above "Number of Entries in Version Number List". This field supports two types of lists, as determined by the Link Id portion of the FDA Address, as follows: Link Id = 0 For linking devices, this is a list of Type 9 live list version numbers, one for each Type 9 interface attached to the linking device. Each element in the list is constructed as follows: Bits 32-17: Type 9 Link Id Bits 16-9: Reserved, set to 0

Parameter name	Octet offset	Data type	Octet length	Description
				Bits 8-1: Version Number The list is ordered by Type 9 bridge interface number with the Live List Version for the first interface occurring first in the list. The value 0 is used for the Link Id bits if the interface has not been assigned a link Id. Link Id > 0 This is a List of Type “X” Node Address Version Numbers, one for each node id currently in the live list for the specified Type “X” link, including the LD Type “X” interface. The List of Type “X” Node Address Version Numbers is used only when this message format is used for SM Identify Responses that describe an LD Type “X” interface. The list begins with the version number of the specified Type “X” Interface. The remainder of the list is sorted in ascending order by Type “X” node address, with two version numbers packed into each Unsigned32 value, as follows: Bits 33-25: Type “X” Node Address Bits 24-17 Version Number Bits 16-9: Type “X” Node Address Bits 8-1: Version Number

4.3.5.4 Type 9 ASE Services

4.3.5.4.1 General note

NOTE Unless specified otherwise, all parameters of the Type 9 Service APDUs and their values are defined in IEC 61158-5 subseries.

4.3.5.4.2 Reject

The Type 9 Reject Service is used only to indicate to the requester that the response is too long for Type 9 to convey it. Instead of defining a separate APDU here to convey this information, the Common Error APDU conveys it, using the Error Class "Reject" and the Reject Code as defined in the Type 9 abstract syntax for the Error Code.

4.3.5.4.3 Initiate (confirmed service Id = 96)

4.3.5.4.3.1 Request APDU parameters

Table 25 defines the request APDU parameters.

Table 25 – Initiate request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Connect Option	0	Unsigned8	1	This field indicates which option is to be used to open the S-VFD Context. Its values are: 1 = VCR Selector 2 = MIB Access 3 = FBAP Access
Access Protection Supported Calling	8	Boolean	1	Service Parameter
Password and Access Groups Calling	2	Unsigned16	2	Service Parameter
Version OD Calling	4	Integer16	2	Service Parameter

Parameter name	Octet offset	Data type	Octet length	Description
Profile Number Calling	6	Unsigned16	2	Service Parameter
PD Tag	8	OctetString	32	The SMK Physical Device Tag of the device containing the destination VFD.

4.3.5.4.3.2 Response APDU parameters

Table 26 lists the response APDU parameters.

Table 26 – Initiate response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Version OD Called	0	Integer16	2	Service Parameter
Profile Number Called	2	Unsigned16	2	Service Parameter

4.3.5.4.3.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1. See Type 9 Initiate specific error codes.

4.3.5.4.4 Abort (unconfirmed service Id = 112)

4.3.5.4.4.1 Request APDU parameters

Table 27 specifies the request APDU parameters.

Table 27 – Abort request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Abort Detail	0	OctetString	16	Supplied by the sender of the APDU.
Abort Identifier	16	Unsigned8	1	Service parameter. Values defined for Type 9 PDUs.
Reason Code	17	Unsigned8	1	Service Parameter
Reserved	18	Unsigned16	2	Reserved, set to 0

4.3.5.4.5 Get status (confirmed service Id = 0)

4.3.5.4.5.1 Request APDU parameters

None.

4.3.5.4.5.2 Response APDU parameters

Table 28 defines the response APDU parameters.

Table 28 – Get response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Logical Status	0	Unsigned8	1	Service Parameter
Physical Status	1	Unsigned8	1	Service Parameter
Reserved	2	Unsigned16	2	Reserved, set to 0
Local Detail	4	OctetString	4	Service Parameter. The high order octet is reserved and set to 0. The low order three octets contain the 3-octet Local Detail value

4.3.5.4.6 Status notification (unconfirmed service Id = 1)

4.3.5.4.6.1 Request APDU parameters

Same as Get Status Response APDU

4.3.5.4.7 Identify (confirmed service Id = 1)

4.3.5.4.7.1 Request APDU parameters

None.

4.3.5.4.7.2 Response APDU parameters

Table 29 specifies the response APDU parameters.

Table 29 – Identify response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Vendor Name	0	VisibleString	32	Service Parameter
Model Identifier	32	VisibleString	32	Service Parameter
Vendor Revision	64	VisibleString	32	Service Parameter

4.3.5.4.8 Get OD (confirmed service Id = 4)

4.3.5.4.8.1 Request APDU parameters

Table 30 specifies the request APDU parameters.

Table 30 – Get OD request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
All Attributes	0	Boolean	1	Service Parameter
Start Index Flag	1	Boolean	1	Service Parameter
Reserved	2	Unsigned16	2	Reserved, set to 0
Index	4	Unsigned32	4	Service Parameter

4.3.5.4.8.2 Response APDU parameters

Table 31 specifies the response APDU parameters.

Table 31 – Get OD response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
More Follows	0	Boolean	1	Service Parameter
Number of Object Descriptions	1	Unsigned8	1	Service Parameter
Reserved	2	Unsigned16	2	Reserved, set to 0
List of Object Descriptions	4	OctetString	N	Defined in 4.3.5.6.2

4.3.5.4.8.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.9 Initiate Put OD (confirmed service Id = 28)

4.3.5.4.9.1 Request APDU parameters

Table 32 specifies the request APDU parameters.

Table 32 – Initiate put OD request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Reserved	0	Unsigned16	2	Reserved, set to 0
Consequence	2	Unsigned16	2	Service Parameter. Values defined for Type 9 PDUs

4.3.5.4.9.2 Response APDU parameters

None

4.3.5.4.9.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.10 Put OD (confirmed service Id = 29)

4.3.5.4.10.1 Request APDU parameters

Table 33 specifies the request APDU parameters.

Table 33 – Put OD request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Number of Object Descriptions	0	Unsigned8	1	Number of entries in List of Object Descriptions
Reserved	1	OctetString	3	Reserved, set to 0
List of Object Descriptions	4	OctetString	N	Defined in 4.3.5.6.2

4.3.5.4.10.2 Response APDU parameters

None

4.3.5.4.10.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.11 Terminate put OD (confirmed service Id = 30)

4.3.5.4.11.1 Request APDU parameters

None.

4.3.5.4.11.2 Response APDU parameters

None.

4.3.5.4.11.3 Error APDU parameters

Defined by the OD Error APDU parameters in 8.3.6.

4.3.5.4.12 Generic initiate download sequence (confirmed service Id = 31)

4.3.5.4.12.1 Service overview

This service corresponds to the Type 9 Initiate Load Service with the following parameter settings:

- Load Region Index Called: Load region of the responder (Server)
- Load Type: DOWNLOAD
- Load Service to Use: Push Segment (Generic Download Segment)
- Load Service Initiator: TRUE (Initiator (Client) initiates the load service (The Push Segment service)).

4.3.5.4.12.2 Request APDU parameters

Table 34 specifies the request APDU parameters.

Table 34 – Generic initiate download sequence request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.12.3 Response APDU parameters

None

4.3.5.4.12.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.13 Generic download segment (confirmed service Id = 32)

4.3.5.4.13.1 Service overview

This service corresponds to the Type 9 Generic Download Segment Service.

4.3.5.4.13.2 Request APDU parameters

See Table 35.

Table 35 – Generic download segment request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
More Follows	4	Boolean	1	Service Parameter
Reserved	5	OctetString	3	Set to 0.
Load Data	8	OctetString	N	Service Parameter

4.3.5.4.13.3 Response APDU parameters

None.

4.3.5.4.13.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.14 Generic terminate download sequence (confirmed service Id = 33)

4.3.5.4.14.1 Service overview

This service corresponds to the Type 9 Generic Terminate Download Sequence Service.

4.3.5.4.14.2 Request APDU parameters

See Table 36.

Table 36 – Generic terminate download sequence request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.14.3 Response APDU parameters

See Table 37.

Table 37 – Response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Terminate Reason	0	Boolean	1	Service Parameter
Reserved	1	OctetString	3	Reserved, set to 0

4.3.5.4.14.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.15 Initiate download sequence (confirmed service Id = 9)

4.3.5.4.15.1 Service overview

This service corresponds to the Type 9 Initiate Download Sequence Service with the following parameter settings:

- Load Region Index Called: Load region of the responder (Server)
- Load Type: DOWNLOAD
- Load Service to Use: Pull Segment (Download Segment)
- Load Service Initiator: FALSE (Remote AP (Server) initiates the load service (The Pull Segment service)).

4.3.5.4.15.2 Request APDU parameters

See Table 38.

Table 38 – Initiate download sequence request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.15.3 Response APDU parameters

None.

4.3.5.4.15.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.16 Download segment (confirmed service Id = 10)

4.3.5.4.16.1 Service overview

This service corresponds to the Type 9 Download Segment Service.

4.3.5.4.16.2 Request APDU parameters

See Table 39.

Table 39 – Download segment request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.16.3 Response APDU parameters

See Table 40.

Table 40 – Download segment response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
More Follows	0	Boolean	1	Service Parameter
Reserved	2	OctetString	3	Reserved, each octet is set to binary 0
Load Data	4	OctetString	N	Service Parameter

4.3.5.4.16.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.17 Terminate download sequence (confirmed service Id = 11)

4.3.5.4.17.1 Service overview

This service corresponds to the Type 9 Terminate Download Sequence Service.

4.3.5.4.17.2 Request APDU parameters

See Table 41.

Table 41 – Terminate download sequence request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Reserved	4	Octetstring	3	Reserved, each octet is set to binary 0
Terminate Reason	7	Boolean	1	Service Parameter

4.3.5.4.17.3 Response APDU parameters

None.

4.3.5.4.17.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.18 Initiate upload sequence (confirmed service Id = 12)

4.3.5.4.18.1 Service overview

This service corresponds to the Type 9 Initiate Upload Sequence Service with the following parameter settings:

- Load Region Index Called: Load region of the responder (Server)
- Load Type: UPLOAD
- Load Service to Use: Pull Segment (Upload Segment)
- Load Service Initiator: TRUE (Initiator (Client) initiates the load service (The Pull Segment service))

4.3.5.4.18.2 Request APDU parameters

See Table 42.

Table 42 – Initiate upload sequence request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.18.3 Response APDU parameters

None.

4.3.5.4.18.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.19 Upload segment (confirmed service Id = 13)

4.3.5.4.19.1 Service overview

This service corresponds to the Type 9 Upload Segment Service.

4.3.5.4.19.2 Request APDU parameters

See Table 43.

Table 43 – Upload segment request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.19.3 Response APDU parameters

See Table 44.

Table 44 – Upload segment response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
More Follows	0	Boolean	1	Service Parameter
Reserved	2	OctetString	3	Reserved, each octet is set to binary 0
Load Data	4	OctetString	N	Service Parameter

4.3.5.4.19.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.20 Terminate upload sequence (confirmed service Id = 14)

4.3.5.4.20.1 Service overview

This service corresponds to the Type 9 Terminate Upload Sequence Service.

4.3.5.4.20.2 Request APDU parameters

See Table 45.

Table 45 – Terminate upload sequence request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.20.3 Response APDU parameters

None

4.3.5.4.20.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.21 Request domain download (confirmed service Id = 15)

4.3.5.4.21.1 Service overview

This service corresponds to the Type 9 Request Domain Download Service.

4.3.5.4.21.2 Request APDU parameters

See Table 46 for definitions.

Table 46 – Request domain download request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Additional Information	4	VisibleString	N	Service Parameter

4.3.5.4.21.3 Response APDU parameters

None.

4.3.5.4.21.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.22 Request domain upload (confirmed service Id = 16)

4.3.5.4.22.1 Service overview

This service corresponds to the Type 9 Create Program Invocation Service.

4.3.5.4.22.2 Request APDU parameters

See Table 47 for definitions.

Table 47 – Request domain upload request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Additional Information	4	VisibleString	N	Service Parameter

4.3.5.4.22.3 Response APDU parameters

None.

4.3.5.4.22.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.23 Create program invocation (confirmed service Id = 17)

4.3.5.4.23.1 Service overview

The Create Program Invocation corresponds to to the Type 9 Create Program Invocation service.

4.3.5.4.23.2 Request APDU parameters

See Table 48 for definitions.

Table 48 – Create program invocation request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Reusable	0	Boolean	1	Service Parameter
Reserved	1	Unsigned8	1	Reserved, set to 0
Number of Domain Indexes	2	Unsigned16	2	The number of domain indexes in this service
List Of Domain Indexes	4	Unsigned32	4 X Number of Domain Indexes	Service Parameter

4.3.5.4.23.3 Response APDU parameters

See Table 49 for definitions.

Table 49 – Create program invocation response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.23.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.24 Delete program invocation (confirmed service Id = 18)

4.3.5.4.24.1 Service overview

The Delete Program Invocation corresponds to to the Type 9 Delete Program Invocation service.

4.3.5.4.24.2 Request APDU parameters

See Table 50 for definitions.

Table 50 – Delete program invocation request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.24.3 Response APDU parameters

None.

4.3.5.4.24.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.25 Start (confirmed service Id = 19)

4.3.5.4.25.1 Request APDU parameters

See Table 51 for definitions.

Table 51 – Start request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Execution Argument	4	OctetString	N	Service Parameter

4.3.5.4.25.2 Response APDU parameters

None.

4.3.5.4.25.3 PI Error APDU parameters

Defined by the Common PI Error APDU parameters in 8.3.6.2.

4.3.5.4.26 Stop (confirmed service Id = 20)

4.3.5.4.26.1 Request APDU parameters

See Table 52 for definitions.

Table 52 – Stop request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.26.2 Response APDU parameters

None.

4.3.5.4.26.3 Error APDU parameters

Defined by the Common PI Error APDU parameters in 8.3.6.2.

4.3.5.4.27 Resume (confirmed service Id = 21)

4.3.5.4.27.1 Request APDU parameters

See Table 53 for definitions.

Table 53 – Resume request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Execution Argument	4	OctetString	N	Service Parameter

4.3.5.4.27.2 Response APDU parameters

None.

4.3.5.4.27.3 Error APDU parameters

Defined by the Common PI Error APDU parameters in 8.3.6.2.

4.3.5.4.28 Reset (confirmed service Id = 22)

4.3.5.4.28.1 Request APDU parameters

See Table 54 for definitions.

Table 54 – Reset request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.28.2 Response APDU parameters

None.

4.3.5.4.28.3 Error APDU parameters

Defined by the Common PI Error APDU parameters in 8.3.6.2.

4.3.5.4.29 Kill (confirmed service Id = 23)

4.3.5.4.29.1 Request APDU parameters

See Table 55 for definitions.

Table 55 – Kill request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.29.2 Response APDU parameters

None.

4.3.5.4.29.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.30 Read (confirmed service Id = 2)

4.3.5.4.30.1 Request APDU parameters

See Table 56 for definitions.

Table 56 – Read request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.30.2 Response APDU parameters

See Table 57 for definitions.

Table 57 – Read response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Value	0	OctetString	N	Service Parameter

4.3.5.4.30.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.31 Read with subindex (confirmed service Id = 82)

4.3.5.4.31.1 Service overview

This service corresponds to the Type 9 Read service.

4.3.5.4.31.2 Request APDU parameters

See Table 58 for definitions.

Table 58 – Read with subindex request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Component ID	4	Unsigned32	4	Service Parameter

4.3.5.4.31.3 Response APDU parameters

See Table 59 for definitions.

Table 59 – Read with subindex response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Value	0	OctetString	N	Service Parameter

4.3.5.4.31.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.32 Write (confirmed service Id = 3)

4.3.5.4.32.1 Request APDU parameters

See Table 60 for definitions.

Table 60 – Write request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Value	4	OctetString	N	Service Parameter

4.3.5.4.32.2 Response APDU parameters

None.

4.3.5.4.32.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.33 Write with subindex (confirmed service Id = 83)

4.3.5.4.33.1 Service overview

This service corresponds to the Type 9 Write service.

4.3.5.4.33.2 Request APDU parameters

See Table 61 for definitions.

Table 61 – Write with subindex request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Component ID	4	Unsigned32	4	Service Parameter
Value	8	OctetString	N	Service Parameter

4.3.5.4.33.3 Response APDU parameters

None.

4.3.5.4.33.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.34 Define variable list (confirmed service Id = 7)

4.3.5.4.34.1 Service overview

This service corresponds to the Type 9 Define Variable List service.

4.3.5.4.34.2 Request APDU parameters

See Table 62 for definitions.

Table 62 – Define variable list request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Number of Indexes	0	Unsigned32	4	Number of Variable Indexes in the list that follows.
List Of Indexes	4	Unsigned32	4 x Number of Indexes	Service Parameter

4.3.5.4.34.3 Response APDU parameters

See Table 63 for definitions.

Table 63 – Define variable list response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.34.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.35 Delete variable list (confirmed service Id = 8)

4.3.5.4.35.1 Service overview

This service corresponds to the Type 9 Delete Variable List service.

4.3.5.4.35.2 Request APDU parameters

See Table 64 for definitions.

Table 64 – Delete variable list request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter

4.3.5.4.35.3 Response APDU parameters

None.

4.3.5.4.35.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.36 Information report (unconfirmed service Id = 0)

4.3.5.4.36.1 Request APDU parameters

See Table 65 for definitions.

Table 65 – Information report request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Value	4	OctetString	N	Service Parameter

4.3.5.4.37 Information report with subindex (unconfirmed service Id = 16)

4.3.5.4.37.1 Service overview

This service corresponds on the Type 9 Information Report service.

4.3.5.4.37.2 Request APDU parameters

See Table 66 for definitions.

Table 66 – Information report with subindex request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Component ID	4	Unsigned32	4	Service Parameter
Value	8	OctetString	N	Service Parameter

4.3.5.4.38 Information report on change (unconfirmed service Id = 17)

4.3.5.4.38.1 Service overview

This service corresponds to the Type 9 Information Report service.

This APDU is used to send data that is published only when its value has changed. The VCR that controls the publishing indicates if this service is to be used.

4.3.5.4.38.2 Request APDU parameters

See Table 67 for definitions.

Table 67 – Information report on change request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Value	4	OctetString	N	Service Parameter

4.3.5.4.39 Information report on change with subindex (unconfirmed service Id = 18)

4.3.5.4.39.1 Service overview

This service corresponds to the Type 9 Information Report service.

This APDU is used to send data that is published only when its value has changed. The VCR that controls the publishing indicates if this service is to be used.

4.3.5.4.39.2 Request APDU parameters

See Table 68 for definitions.

Table 68 – Information report on change with subindex request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Component ID	4	Unsigned32	4	Service Parameter
Value	8	OctetString	N	Service Parameter

4.3.5.4.40 Event notification (unconfirmed service Id = 2)

4.3.5.4.40.1 Request APDU parameters

See Table 69 for definitions.

Table 69 – Event notification request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Event Number	4	Unsigned32	4	Service Parameter
Data	8	OctetString	N	Service Parameter

4.3.5.4.41 Alter event condition monitoring (confirmed service Id = 24)

4.3.5.4.41.1 Request APDU parameters

See Table 70 for definitions.

Table 70 – Alter event condition monitoring request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Reserved	4	Octetstring	3	Reserved, each octet is set to binary 0
Enabled	7	Boolean	1	Service Parameter

4.3.5.4.41.2 Response APDU parameters

None.

4.3.5.4.41.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.4.42 Acknowledge event notification (confirmed service Id = 25)

4.3.5.4.42.1 Request APDU parameters

See Table 71 for definitions.

Table 71 – Acknowledge event notification request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Service Parameter
Event Number	4	Unsigned32	4	Service Parameter

4.3.5.4.42.2 Response APDU parameters

None.

4.3.5.4.42.3 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.5 LAN redundancy services

4.3.5.5.1 LAN redundancy diagnostic message (unconfirmed service Id = 1)

4.3.5.5.1.1 Request APDU parameters

See Table 72 for definitions.

Table 72 – LAN redundancy diagnostic message request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Device Index	0	Unsigned16	2	Service parameter 0 = Index not assigned 1 – 65535 = Device Index
Number of Interfaces	2	Unsigned8	1	Service parameter.
Transmission Interface	3	Unsigned8	1	Identifies the interface from which this APDU was transmitted 1 = Interface A 2 = Interface B
Diagnostic Message Interval	4	Unsigned 32	4	Service parameter.
SMK Physical Device Tag	8	VisibleString	32	Service parameter. Unused octets in this field are set to ASCII space characters.
Reserved	40	Unsigned8	1	Reserved. Set to zero
Duplicate Detection State	41	Unsigned8	1	Service parameter. Encoded as follows: Bits 8-3: Reserved, set to 0. Bit 2: 1 = Duplicate PD Tag Detected 0 = Duplicate PD Tag Not Detected Bit 1: 1 = Duplicate Device Index Detected 0 = Duplicate Device Index Not Detected
Number of Interface	42	Unsigned16	2	The number of Unsigned32 entries in the four Array of Interface Statuses that follow. The value of this

Parameter name	Octet offset	Data type	Octet length	Description
Statuses				field is 1/32nd of the Max Device Index value configured using the SM Set Assignment service.
Array of Interface AtoA Statuses	44	Array of Unsigned32	4* Number of Interface Statuses	Each 32-bit value represents the status of 32 devices. (see note) 1 bit represents each device. Each bit indicates the status of the transmission path from interface A of the reporting device to interface A of this device. The following values are used for each bit: 0 = Diagnostic messages successfully 1 = Diagnostic messages not received
Array of Interface BtoA Statuses	44 + 4* Number of Interface Statuses	Array of Unsigned32	4* Number of Interface Statuses	Each 32-bit value represents the status of 32 devices. (see note) 1 bit represents each device. Each bit indicates the status of the transmission path from interface B of the reporting device to interface A of this device. The following values are used for each bit: 0 = Diagnostic messages successfully 1 = Diagnostic messages not received
Array of Interface AtoB Statuses	44 + 8* Number of Interface Statuses	Array of Unsigned32	4* Number of Interface Statuses	Each 32-bit value represents the status of 32 devices. (see note) 1 bit represents each device. Each bit indicates the status of the transmission path from interface A of the reporting device to interface B of this device. The following values are used for each bit: 0 = Diagnostic messages successfully 1 = Diagnostic messages not received
Array of Interface BtoB Statuses	44 + 12* Number of Interface Statuses	Array of Unsigned32	4* Number of Interface Statuses	Each 32-bit value represents the status of 32 devices. (see note) 1 bit represents each device. Each bit indicates the status of the transmission path from interface B of the reporting device to interface B of this device. The following values are used for each bit: 0 = Diagnostic messages successfully 1 = Diagnostic messages not received

NOTE The bit that identifies a device is located by its bit position in the array. Dividing the device's Device Index by 32 yields the index of the zero-based Unsigned32 array element. The remainder from the division identifies the specific bit in the Unsigned32 array element, counting from the most significant bit (= remainder 1). Therefore, a device with Device Index 33 would be represented by the most significant bit (bit 32) of the second Unsigned32 element of the array, and a device with Device Index 34 would be represented by the next most significant bit (bit 31) of the second Unsigned32 element of the array.

4.3.5.5.2 LAN redundancy get information (confirmed service Id = 1)

4.3.5.5.2.1 Service overview

This service is always confirmed and, therefore, always uses the Invoke Id in the Message Trailer.

4.3.5.5.2.2 Request APDU parameters

None.

4.3.5.5.2.3 Response APDU parameters

See Table 73 for definitions.

Table 73 – LAN redundancy get information response APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
LAN Redundancy Attributes Version	0	Unsigned32	4	Service Parameter
Max Sequence Number Difference	4	Unsigned8	1	Service Parameter
LAN Redundancy Flags	5	Unsigned8	1	Service Parameter. Encoded as follows; 0 = False, 1 = True Bit 1 Single Mcast APDU Tx I/F Enabled Bit 2 Crossed Cable Detection Enabled Bit 3 Single Mcast APDU Rcpt I/F Enabled Bit 4 Diagnosis using own APDUs Enabled Bit 5 Load Balancing Enabled Bits 6-8 Reserved, set to 0
Reserved	6	Unsigned16	2	Reserved. Set to zero.
Diagnostic Message Interval	8	Unsigned32	4	Service Parameter
AgingTime	12	Unsigned32	4	Service Parameter
Diagnostic Message Interface A Send Address	16	OctetString	16	This parameter is part of the Diagnostic Message Send Addresses service parameter. It defines the destination address for diagnostic messages sent from interface A.
Diagnostic Message Interface A Receive Address	32	OctetString	16	This parameter is part of the Diagnostic Message Receive Addresses service parameter. It defines the destination address for diagnostic messages received on interface A.
Diagnostic Message Interface B Send Address	48	OctetString	16	This parameter is part of the Diagnostic Message Send Addresses service parameter. It defines the destination address for diagnostic messages sent from interface B.
Diagnostic Message Interface B Receive Address	64	OctetString	16	This parameter is part of the Diagnostic Message Receive Addresses service parameter. It defines the destination address for diagnostic messages received on interface B.

4.3.5.5.2.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.5.3 LAN redundancy put information (confirmed service Id = 2)

4.3.5.5.3.1 Service overview

This service is always confirmed and, therefore, always uses the Invoke Id in the Message Trailer.

4.3.5.5.3.2 Request APDU parameters

Same as LAN Redundancy Get Information response parameters. Only those fields following the APDU Version Number are used by the receiving LAN Redundancy Entity.

4.3.5.5.3.3 Response APDU parameters

Same as Request Message parameters. It includes the updated LAN Redundancy Attributes Version.

4.3.5.5.3.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.5.4 LAN redundancy get statistics (confirmed service Id = 3)

4.3.5.5.4.1 Service overview

This service is always confirmed and, therefore, always uses the Invoke Id in the Message Trailer.

4.3.5.5.4.2 Request APDU parameters

None.

4.3.5.5.4.3 Response APDU parameters

See Table 74 for definitions.

Table 74 – LAN redundancy get statistics request APDU parameters

Parameter name	Octet offset	Data type	Octet length	Description
Diagnostic Messages Received On Interface A	0	Unsigned32	4	Service Parameter
Diagnostic Messages Missed On Interface A	4	Unsigned32	4	Service Parameter
Remote Device Diagnostic Message Receive Faults Detected On Interface A	8	Unsigned32	4	Service Parameter
Diagnostic Messages Received On Interface B	12	Unsigned32	4	Service Parameter
Diagnostic Messages Missed On Interface B	16	Unsigned32	4	Service Parameter
Remote Device Diagnostic Message Receive Faults Detected On Interface B	20	Unsigned32	4	Service Parameter
Number of Crossed Cable	24	Unsigned32	4	This field is number of Unsigned32 values in the List of Crossed Cable Status. 0 means that the list is not

Parameter name	Octet offset	Data type	Octet length	Description
Statuses				present. If the list is present, then this field contains 1/32nd of the value of the Max Device Index assigned to the device in the SM Set Assignment Info message.
Array of Crossed Cable Status	28	Array of Unsigned32	4 * Number of Crossed Cable Statuses	Service Parameter. Each bit in this array (see note) represents a device that sends diagnostic messages to the device that is sending this message (the reporting device). Each remote device's bit represents whether the interface it is using to send diagnostic messages, as indicated by the Interface Used for Transmission of this Diagnostic Message field, is the same interface used by the reporting device to receive the messages. The following values are used for each bit: 0 = Cables not crossed or Crossed Cable Detection Enabled is false. 1 = Cables are crossed (Diagnostic message sent from Interface A was received on Interface B, or diagnostic message sent from Interface B was received on Interface A).

NOTE The bit that identifies a device is located by its bit position in the array. Dividing the device's Device Index by 32 yields the index of the zero-based Unsigned32 array element. The remainder from the division identifies the specific bit in the Unsigned32 array element, counting from the most significant bit (= remainder 1). Therefore, a device with Device Index 33 would be represented by the most significant bit (bit 32) of the second Unsigned32 element of the array, and a device with Device Index 34 would be represented by the next most significant bit (bit 31) of the second Unsigned32 element of the array.

4.3.5.5.4.4 Error APDU parameters

Defined by the Common Error APDU parameters in 8.3.6.1.

4.3.5.6 Structure of object descriptions

4.3.5.6.1 General

Subclause 4.3.5.6 defines the object descriptions conveyed using Type 5 Get OD and Put OD services. VFDs using 32-bit OD indexes are required to use the HSE object description formats defined below. VFDs using 16-bit OD indexes may use either H1 or HSE object description formats.

The FDA Agent may convert Type 9 formatted object descriptions it receives from local VFDs or from Type 9 devices to the corresponding Type 5 formats, but it is not required to do so. In addition, the FDA Agent may, but is not required to, convert HSE formatted object descriptions that it receives from Type 5 PutOD request messages to Type 9 format for delivery to or forwarding to VFDs that use Type 9 formats.

When an FDA Agent that does not support conversion from Type 5 to Type 9 format receives such a request message requiring conversion, it may simply forward the message and rely on the receiving VFD to return the appropriate error message, or it may return a negative response with error class "od" and error code ="type 5 to type 9 format conversion not supported".

The fields contained in each OD construct below are defined in the Type 9 Specification.

Their definitions are the same here, however, their *index*, *data type,* and *length* fields have changed to 32-bits, and their order have changed to align on 32-bit boundaries.

For fields defined to have Boolean values, the value 0 indicates "FALSE" and any non-zero value indicates "TRUE".

To maintain compatibility with Type 9 object descriptions, the first field in an object description conveyed by the FDA Agent is the OD Tag. This OD Tag contains either the tag of a Type 9 object description or a tag reserved for Type 5 object descriptions (hex value 0xFFFF). Following the OD Tag is the object description indicated by the tag. Each Type 5 object description uses one of the formats specified in 4.3.5.6.

Each Type 9 object description contains a Type 5 tag plus a Type 5 object description, as specified by Type 9. The Type 5 tag is the encoded Type 9 ASN.1 tag specified for the GetOD Response and the PutOD Request messages. The Type 9 Object Description is specified as packed data.

The Type 9 tag consists of one or two octets. The first four most significant bits of the first octet (most significant byte) has the value 0. The next four bits is the length of the object description. If this 4-bit length field contains a value less than the hex value 0xF, then the H1 tag is one octet and the 4-bit length field specifies the length of the object description that follows.

Following the Type 5 Type Flag field, all Type 5 Object Descriptions have five additional fields. The format of this common Type 5 Object Description header is shown in Table 75.

Table 75 – Object description header

Field name	Octet offset	Data type	Octet length	Description
OD Tag	0	Unsigned16	2	Set to $FFFF_{16}$ to indicate an Type 5 Object Description
Object Code	2	Integer8	1	Object Code of the Object
All Attributes Flag	3	Unsigned8	1	Indicates long form (=1) or short form (=0) Object Description
Object Description Length	4	Unsigned16	2	Total length in octets of the Object Description
Reserved (set to zero)	6	Unsigned16	2	Set to zero
Index32	8	Unsigned32	4	32-bit Object Index

A null object in Type 5 is used to indicate an unused object index, as it is in Type 9. It is represent in Type 5 as shown in Table 76.

Table 76 – Null object

Field name	Value
OD Tag	$FF\ FF_{16}$
Object Code	0
All Attributes Flag	0
Object Description Length	12
Reserved (set to zero)	0
Index32	Index of Null Object

4.3.5.6.2 Structure of the list of object descriptions

See Table 77 for definitions.

Table 77 – Structure of the list of object descriptions

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 16	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
ROM/RAM Flag	12	Unsigned8	1
Access Protection Supported	13	Unsigned8	1
Reserved (set to zero)	14	Unsigned16	2
Name Length	16	Unsigned32	4
Version OD	20	Unsigned32	4
ST-OD Length	24	Unsigned32	4
First Index S-OD	28	Unsigned32	4
S-OD Length	32	Unsigned32	4
First Index DV-OD	36	Unsigned32	4
DV-OD Length	40	Unsigned32	4
First Index DP-OD	44	Unsigned32	4
DP-OD Length	48	Unsigned32	4
Long Form Additional Fields (see note)	52		
Local Address OD-ODES	0 + Long Form Offset	Unsigned64	8
Local Address ST-OD	8 + Long Form Offset	Unsigned64	8
Local Address S-OD	16 + Long Form Offset	Unsigned64	8
Local Address DV-OD	24 + Long Form Offset	Unsigned64	8
Local Address DP-OD	32 + Long Form Offset	Unsigned64	8

NOTE If the Short Form is requested, these fields are not present.

4.3.5.6.3 Structure of a load region in the S-OD

See Table 78 for definitions.

Table 78 – Structure of a load region in the S-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 18	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Domain State	12	Unsigned8	1
Upload State	13	Unsigned8	1
Reserved (set to zero)	14	Unsigned16	2

Field name	Octet offset	Data type	Octet length
Max Octets	16	Unsigned32	4
Counter	20	Unsigned32	4
Long Form Additional Fields (see note 1)	24		
Local Address	0 + Long Form Offset	Unsigned64	8
Password	8 + Long Form Offset	Unsigned8	1
Access Groups	9 + Long Form Offset	Unsigned8	1
Access Rights	10 + Long Form Offset	Unsigned16	2
Extension Length	12 + Long Form Offset	Unsigned32	4
Extension Data	16 + Long Form Offset	OctetString	Extension Length
Domain Name	16 + Long Form Offset + Extension Length	VisibleString	Name Length (see note 2)

NOTE 1 If the Short Form is requested, these fields are not present.

NOTE 2 If Name Length in the OD Object Description is 0, then this field is not present.

4.3.5.6.4 Structure of a function invocation in the DP-OD

See Table 79 for definitions.

Table 79 – Structure of a function invocation in the DP-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 19	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Deletable	12	Unsigned8	1
Reusable	13	Unsigned8	1
PI-State	14	Unsigned8	1
Reserved	15	Unsigned8	3
Number Of Domains	16	Unsigned32	4
List of Domain Indexes	20	Unsigned32	Number Of Domains x 4 (see note 1)
Long Form Additional Fields (see note 2)	20 + (Number Of Domains x 4)		
Password	0 + Long Form Offset	Unsigned8	1
Access Groups	1 + Long Form Offset	Unsigned8	1
Access Rights	2 + Long Form Offset	Unsigned16	2
Extension Length	4 + Long Form Offset	Unsigned32	4
Extension Data	8 + Long Form Offset	OctetString	Extension Length
PI Name	8 + Long Form Offset + Extension Length	VisibleString	Name Length (see note 3)

NOTE 1 If Number Of Domains is 0, then this field is not present.

NOTE 2 If the Short Form is requested, these fields are not present.

NOTE 3 If Name Length in the OD Object Description is 0, then this field is not present.

4.3.5.6.5 Structure of an event in the S-OD

See Table 80 for definitions.

Table 80 – Structure of an event in the S-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 20	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Reserved	12	OctetString	3
Enabled	15	Unsigned8	1
Index Event Data	16	Unsigned32	4
Length	20	Unsigned32	4
Long Form Additional Fields (see note 1)	24		
Password	0 + Long Form Offset	Unsigned8	1
Access Groups	1 + Long Form Offset	Unsigned8	1
Access Rights	2 + Long Form Offset	Unsigned16	2
Extension Length	4 + Long Form Offset	Unsigned32	4
Extension Data	8 + Long Form Offset	OctetString	Extension Length
Event Name	8 + Long Form Offset + Extension Length	VisibleString	Name Length (see note 2)

NOTE 1 If the Short Form is requested, these fields are not present.

NOTE 2 If Name Length in the OD Object Description is 0, then this field is not present.

4.3.5.6.6 Structure of a data type in the ST-OD

See Table 81 for definitions.

Table 81 – Structure of a data type in the ST-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 21	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Reserved	10	Unsigned16	2
Length of Data Type	12	Unsigned32	4
Symbolic Name Length	16	Unsigned32	4
*Symbolic Name	20	VisibleString	Symbolic Name Length

NOTE If the Symbolic Name Length is 0, then this field is not present.

4.3.5.6.7 Structure of a data type structure description in the ST-OD

See Table 82 for definitions.

Table 82 – Structure of a data type structure description in the ST-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 22	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Reserved (set to zero)	12	Unsigned16	2
Number of Elements	14	Unsigned16	2
List of Elements	16	Structure	8 x Number of Elements
Data Type Index		Unsigned32	4
Length		Unsigned32	4

4.3.5.6.8 Structure of a simple variable in the S-OD

See Table 83 for definitions.

Table 83 – Structure of a simple variable in the S-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 23	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Data Type Index	12	Unsigned32	4
Length	16	Unsigned32	4
Long Form Additional Fields (see note 1)	20		
Local Address	0 + Long Form Offset	Unsigned64	8
Password	8 + Long Form Offset	Unsigned8	1
Access Groups	9 + Long Form Offset	Unsigned8	1
Access Rights	10 + Long Form Offset	Unsigned16	2
Extension Length	12 + Long Form Offset	Unsigned32	4
Extension Data	16 + Long Form Offset	OctetString	Extension Length
Variable Name	16 + Long Form Offset + Extension Length	VisibleString	Name Length (see note 2)

NOTE 1 If the Short Form is requested, these fields are not present.

NOTE 2 If Name Length in the OD Object Description is 0, then this field is not present.

4.3.5.6.9 Structure of an array in the S-OD

See Table 84 for definitions.

Table 84 – Structure of an array in the S-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 24	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Number of Elements	12	Unsigned32	2
Data Type Index	16	Unsigned32	4
Length	20	Unsigned32	4
Long Form Additional Fields (see note 1)	24		
Local Address	0 + Long Form Offset	Unsigned64	8
Password	8 + Long Form Offset	Unsigned8	1
Access Groups	9 + Long Form Offset	Unsigned8	1
Access Rights	10 + Long Form Offset	Unsigned16	2
Extension Length	12 + Long Form Offset	Unsigned32	4
Extension Data	16 + Long Form Offset	OctetString	Extension Length
Variable Name	16 + Long Form Offset + Extension Length	VisibleString	Name Length (see note 2)

NOTE 1 If the Short Form is requested, these fields are not present.

NOTE 2 If Name Length in the OD Object Description is 0, then this field is not present.

4.3.5.6.10 Structure of a record in the S-OD

See Table 85 for definitions.

Table 85 – Structure of a record in the S-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 25	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Data Type Index	12	Unsigned32	4
Reserved	16	Unsigned32	4
Long Form Additional Fields (see note 1)	20		
List of Local Addresses	0 + Long Form Offset	Unsigned64	K x 8

Field name	Octet offset	Data type	Octet length
			(see note 2)
Password	(K x 8) + Long Form Offset	Unsigned8	1
Access Groups	1 + (K x 8) + Long Form Offset	Unsigned8	1
Access Rights	2 + (K x 8) + Long Form Offset	Unsigned16	2
Extension Length	4 + (K x 8) + Long Form Offset	Unsigned32	4
Extension Data	8 + (K x 8) + Long Form Offset	OctetString	Extension Length
Record Name	8 + (K x 8) + Long Form Offset + Extension Length	VisibleString	Name Length (see note 3)

NOTE 1 If the Short Form is requested, these fields are not present.

NOTE 2 K represents the number of fields in the record.

NOTE 3 If Name Length in the OD Object Description is 0, then this field is not present.

4.3.5.6.11 Structure of a variable list in the DV-OD

See Table 86 for definitions.

Table 86 – Structure of a variable list in the DV-OD

Field name	Octet offset	Data type	Octet length
Short Form Fields			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 26	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (set to zero)	6	Unsigned16	2
Index32	8	Unsigned32	4
Deletable	12	Unsigned8	1
Reserved	13	Unsigned8	1
Number of Elements	14	Unsigned16	2
List of Element Indexes	16	Unsigned32	4 x Number of Elements
Long Form Additional Fields (see note 1)	20 + (K x 4) (see note 2)		
Password	0 + Long Form Offset	Unsigned8	1
Access Groups	1 + Long Form Offset	Unsigned8	1
Access Rights	2 + Long Form Offset	Unsigned16	2
Extension Length	4 + Long Form Offset	Unsigned32	4
Extension Data	8 + Long Form Offset	OctetString	Extension Length
Variable List Name	8 + Long Form Offset + Extension Length	VisibleString	Name Length (see note 3)

NOTE 1 If the Short Form is requested, these fields are not present.

NOTE 2 K represents the Number of Elements in the record.

NOTE 3 If Name Length in the OD Object Description is 0, then this field is not present.

4.3.6 Error APDU bodies

4.3.6.1 Common error parameters

The following error parameter format as shown in Table 87 is used for many services.

Table 87 – Common error parameters

Parameter name	Octet offset	Data type	Octet length	Description
Error Class	0	Unsigned8	1	This parameter identifies the category of the error. Its values are defined in the Error Class production of the Type 9 Abstract Syntax. The values are the tagged values. The value 9 has been added for the Type 5 Initiate Error APDU use.
Error Code	1	Unsigned8	1	This parameter identifies the specific type of error within the class Its values are defined in the Error Class production of the Type 9 Abstract Syntax. The values are the enumerated values with parenthesized values, such as other = 0. Values for error class 11 have been added for Initiate Error message use.
Additional Code	3	Integer16	1	This parameter provides an additional code that is associated with the error.
Additional Description	4	VisibleString	16	This parameter provides 16 octets of additional information that is associated with the error.

4.3.6.2 PI error parameters

See Table 88 for definitions.

Table 88 – PI error parameters

Parameter name	Octet offset	Data type	Octet length	Description
Pi State	0	Unsigned8	1	Defined in Service Parameter.
Reserved	1	OctetString	3	Reserved, set to 0
Error Class	4	Unsigned8	1	Defined in Service Parameter.
Error Code	5	Unsigned8	1	Defined in Service Parameter.
Additional Code	7	Integer16	1	Defined in Service Parameter.
Additional Description	8	VisibleString	16	Defined in Service Parameter.

4.3.6.3 OD error parameters

See Table 89 for definitions.

Table 89 – OD error parameters

Parameter name	Octet offset	Data type	Octet length	Description
Index	0	Unsigned32	4	Defined in the Service Parameter.
Error Class	4	Unsigned8	1	Defined in the Service Parameter.
Error Code	5	Unsigned8	1	Defined in the Service Parameter.
Additional Code	7	Integer16	1	Defined in the Service Parameter.
Additional Description	8	VisibleString	16	Defined in the Service Parameter.

4.3.6.4 Error class and error code values

Table 90 specifies the FDA Agent error codes for each error class.

Table 90 – Error class and error code values

Description:		Values:	
Class	Code	Class	Code
vfd state	other	1	0
application reference	other	2	0
	application unreachable	2	1
definition	other	3	0
	object undefined	3	1
	object attributes inconsistent	3	2
	name already exists	3	3
resource	other	4	0
	memory unavailable	4	1
	max outstanding requests per session exceeded	4	2
	max sessions exceeded	4	3
	object creation failure	4	4
service	other	5	0
	object state conflict	5	1
	pdu size	5	2
	object constraint conflict	5	3
	parameter inconsistent	5	4
	illegal parameter	5	5
	unsupported service	5	6
	unsupported version	5	7
	invalid options	5	8
	unsupported protocol	5	9
	reserved	5	10
	key parameter mismatch	5	11
	assignments already made	5	12
	unsupported device redundancy state	5	13
	response time-out	5	14
	duplicate PD Tag detected	5	15
access	other	6	0
	object invalidated	6	1
	hardware fault	6	2
	object access denied	6	3
	invalid address	6	4
	object attribute inconsistent	6	5
	object access unsupported	6	6
	object non existent	6	7
	type conflict	6	8

Description:		Values:	
Class	Code	Class	Code
	named access unsupported	6	9
	access to element unsupported	6	10
	config access already open	6	11
	reserved	6	12
	unrecognized FDA Address	6	13
od	other	7	0
	name length overflow	7	1
	od overflow	7	2
	od write protected	7	3
	extension length overflow	7	4
	od description length overflow	7	5
	operational problem	7	6
	type 5 to type 9 format conversion not supported	7	7
other	other	8	0
reject	pdu size	9	5
sm reason code	other	10	0
	DLL Error – insufficient resources	10	1
	DLL Error – sending queue full	10	2
	DLL Error – time-out before transmission	10	3
	DLL Error – reason unspecified	10	4
	Device failed to respond to SET_PD_TAG	10	5
	Device failed to respond to WHO_HAS_PD_TAG	10	6
	Device failed to respond to SET_ADDR	10	7
	Device failed to respond to IDENTIFY	10	8
	Device failed to respond to ENABLE_SM_OP	10	9
	Device failed to respond to CLEAR_ADDRESS	10	10
	Multiple Response from WHO_HAS_PD_TAG	10	11
	Non-Matching PD_TAG from WHO_HAS_PD_TAG	10	12
	Non-Matching PD_TAG from IDENTIFY	10	13
	Non-Matching DEV_ID from IDENTIFY	10	14
	Remote Error Invalid State	10	15
	Remote Error PD-Tag doesn’t match	10	16
	Remote Error Dev-ID doesn’t match	10	17
	Remote Error SMIB object write failed	10	18
	Remote Error Starting SM Operational	10	19
initiate	other	11	0
	max-fms-pdu-size-insufficient	11	1
	feature-not-supported	11	2
	version-od-incompatible	11	3
	user-initiate-denied	11	4
	password-error	11	5
	profile-number-incompatible	11	6

4.4 FAL protocol state machine structure

The FAL Protocol state machine structure defined for Type 9 applies.

4.5 SMK state machine

4.5.1 General

The SMK protocol machine (SMKPM) defines the behavior of the SMK for those states that affect its communications.

In the protocol machine that follows:

1. The Invoke Id parameter is not included explicitly. Instead, it is considered to be an integral part of all confirmed service and Find Tag primitives. It is received from the sending SMK user and conveyed implicitly to the remote SMK user.
2. The response address is not included explicitly. It is always taken from the source network address of the request.
3. To send and receive SMK APDUs, ARs are not used. Instead, the SMKPM performs the necessary AR functions.
4. The double forward slash, i.e. "//" marks the beginning of a comment in a line.

4.5.2 Messages received by the SMKPM

The SMKPM receives APDUs from the FDA Agent. These APDUs are modeled as Events of SMKPM transitions. The RcvMsg() function is used to represent the arrival of an APDU.

4.5.3 SMKPM service primitives and local APs, Type 5 SMK service processors, and Type "X" interface functions

Table 91 shows the service primitives (in alphabetic order) used in the SMKPM. They are used as follows.

- Request service primitives are used as Events of SMKPM transitions. They are received from Local APs.
- Request service primitives are used also in Actions of SMKPM transitions. When used in this fashion, they cause the corresponding event to be delivered to the SMKPM and are processed immediately, ahead of all other queued events.
- Indication service primitives are used in Actions of SMKPM transtions. They may be delivered to Local APs or to Type "X" Interface Functions. There is one interface function for each Type "X" interface in a linking device. Each is identified by the link id of the FDA Address contained in the received APDU. Link Id 00 00 when used with the Type "X" NN.SS SMK group address means all Type "X" Interface Functions. Service names preceded by "LD_" identify services provided by Type "X" Interface Functions. These services take the sm_svc parameter of the received APDU as a parameter. They invoke one or more Type "X" SM services to satisfy requests received from the Type 5 SMK. Issuing one of these service requests causes the Type 5 SMK to save the Invoke Id so that it can match the response with the request. The Type "X" services used and their sequence is implementation dependent. If the Type "X" Interface Function receives an LD_FindTagQuery request and locates the queried object, it responds by issuing an SM_FindTagReply request.
- Response service primitives are used as Events of SMKPM transitions. They may be received from a Local AP or from an Interface Functions.

- Response service primitives are used also in Actions of SMKPM transitions. When used in this fashion, they cause the corresponding event to be delivered to the SMKPM and are processed immediately, ahead of all other queued events.
- Confirmation service primitives are used in Actions of SMKPM transtions. They are delivered to the requesting Local APs.

Table 91 – SMKPM service primitives

Service	Primitives			
	.req	.ind	.rsp[a]	.cnf
Type_X_ClearAddress		X	X	
Type_X_ClearAssignmentInfo		X	X	
Type_X_FindTagQuery		X		
Type_X_FindTagReply		X		
Type_X_SetAssignmentInfo		X	X	
SM_ClearAddress	X	X	X	X
SM_ClearAssignmentInfo	X	X	X	X
SM_DeviceAnnunciation	X			
SM_FindTagQuery	X	X		
SM_FindTagReply	X	X		
SM_Identify	X	X	X	X
SM_SetAssignmentInfo	X	X	X	X

[a]. rsp is the positive response primitive as well as the negative response primitive.

4.5.4 State table transition processing

The state tables that follow are composed of a series of numbered transitions. The order of processing the transitions is based on the following rules.

- When receiving an event, locate the first transition in the table containing an instance of the event. A communications event is identified by its indication, e.g. SM_Clear Address.ind.
- Beginning with that transition, proceed down the state table, transition by transition, until a transition is located in which its Current state matches the current state of the state machine and its Event matches the received event.
- Test the conditions. If the result is false, proceed down to the next transition that matches the Current state and Event.
- If no transition matches, then ignore the event.
- When a transition is located, perform the specified action and change the current state of the state machine to the Next state specified by the transition. Then wait for the next event.

4.5.5 States

Table 92 and Figure 1 define the states and their transitions.

Table 92 – SMKPM states

No Address	The device does not have an network address on any of its Type 5 interfaces. If it has an network address on any of its Type 5 interfaces, it is not in this state. In this state, the SMKPM waits for an event that receives an network address through a local interface from the underlying layers.
No Tag	The device does not have valid assignment info. In this state, the SMK sends Annunciate requests and listens for a valid Set Assignment Info APDU. It also responds to Clear Address and SM Identify Request APDUs.
Operational	The device has an network address and valid assignment info. This is the normal operating state of the SMK.

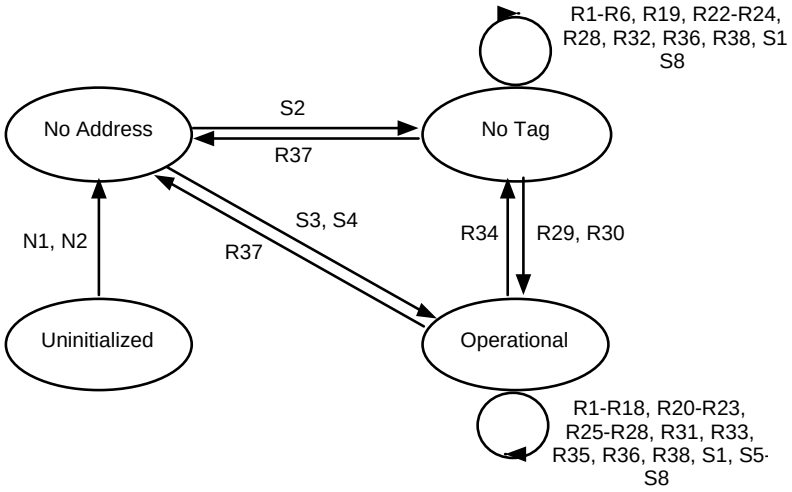


Figure 1 – State transition diagram for SMK

4.5.6 State tables

Table 93 through Table 95 define the state machine.

Table 93 – SMKPM state table – initialization

#	Current state	Event or condition => action	Next state
6) N1	7) Uninitialized	8) Powerup 9) && OperationalRestore () = "false" 10) => 11) RestoreDefaults ()	12) No Address
13) N2	14) Uninitialized	15) Powerup 16) && OperationalRestore () = "true" 17) => 18) // No action	19) No Address

Table 94 – SMKPM state table – receive transitions

#	Current state	Event or condition => action	Next state
20) R1	21) Operational 22) No Tag	23) RcvMsg() = "Any Unconfirmed SM Req Message" 24) && FdaAddressType (sm_svc) = "UNRECOGNIZED" 25) => 26) // Do nothing – no response for unconfirmed messages	27) Same
28) R2	29) Operational 30) No Tag	31) RcvMsg() = "Any Confirmed SM Req Message" 32) && FdaAddressType (sm_svc) = "UNRECOGNIZED" 33) => 34) SM_ConfirmedService.err { 35) sm_service_type = SvcType (sm_svc), 36) error_class = "access" 37) error_code = "unrecognized FDA Address" 38) addl_code = 2 39) }	40) Same
41) R3	42) Operational 43) No Tag	44) RcvMsg() = "Any Confirmed SM Req Message" 45) && FdaAddressType (sm_svc) = "GROUP SMK" 46) => 47) SM_ConfirmedService.err { 48) sm_service_type = SvcType (sm_svc), 49) error_class = "access" 50) error_code = "unrecognized FDA Address" 51) addl_code = 3 52) }	53) Same
54) R4	55) Operational 56) No Tag	57) RcvMsg() = "Any Confirmed SM Req Message" 58) && IsValid (sm_svc) = "false" 59) => 60) SM_ConfirmedService.err { 61) sm_service_type = SvcType (sm_svc), 62) error_class = "service" 63) error_code = "parameter inconsistent" 64) addl_code = GetAddlCode () 65) }	66) Same
67) R5	68) Operational 69) No Tag	70) RcvMsg() = "Any Confirmed SM Rsp Message" || 71) RcvMsg() = "Any Confirmed SM Error Message" 72) => 73) SM_ConfirmedService.cnf {} * 74) 75) * How the SMKPM locates the requesting AP is not specified. There may be more than one way to do this. Error checking is performed on the response by the requesting AP.	76) Same
77) R6	78) Operational 79) No Tag	80) RcvMsg() = "SM_FindTagQuery" 81) && IsValid (sm_svc) = “false” 82) => 83) // Do nothing – no response for unconfirmed messages	84) Same
85) R7	86) Operational	87) RcvMsg() = "SM_FindTagQuery" 88) && (FdaAddressType (sm_svc) = "HSE SMK" 89) || FdaAddressType (sm_svc) = "GROUP SMK") 90) && DeviceRedundancyState () = "secondary" 91) && (QueryType (sm_svc) = “Secondary PD Tag” 92) || QueryType (sm_svc) = “Device Index”) 93) && QueryMatch (sm_svc) = "true" 94) => 95) SM_FindTagReply.req {}	96) Same

97) R8	98) Operational	99) RcvMsg() = "SM_FindTagQuery"
		100) && DeviceRedundancyState () = "secondary"
		101) =>
		102) // Discard message and do nothing. Secondaries do not respond to
		103) // any other queries
			104) Same

105) R9	106) Operational	107) RcvMsg() = "SM_FindTagQuery" 108) && (FdaAddressType (sm_svc) = "HSE SMK" 109) || FdaAddressType (sm_svc) = "GROUP SMK") 110) && (QueryType (sm_svc) = "Device Index" 111) || QueryType (sm_svc) = "VFD Reference") 112) && QueryMatch (sm_svc) = "true" 113) => 114) SM_FindTagReply.req {}	115) Same
116) R10	117) Operational	118) RcvMsg() = "SM_FindTagQuery" 119) && FdaAddressType (sm_svc) = "HSE SMK"	125) Same

#	Current state	**Event** or **condition** **=> action**	Next state
		120) && (QueryType (sm_svc) = “Primary PD Tag” 121) || QueryType (sm_svc) = “VFD Tag”) 122) && QueryMatch (sm_svc) = "true" 123) => 124) SM_FindTagReply.req {}	
126) R11	127) Operational	128) RcvMsg() = "SM_FindTagQuery" 129) && FdaAddressType (sm_svc) = "GROUP SMK" 130) && (QueryType (sm_svc) = “Primary PD Tag” || 131) QueryType (sm_svc) = “VFD Tag”) 132) && QueryMatch (sm_svc) = "true" 133) => 134) SM_FindTagReply.req {} 135) Type_X_FindTagQuery.ind {}	136) Same
137) R12	138) Operational	139) RcvMsg() = "SM_FindTagQuery" 140) && FdaAddressType (sm_svc) = "GROUP SMK" 141) && (QueryType (sm_svc) = “Primary PD Tag” 142) || QueryType (sm_svc) = “VFD Tag”) 143) && QueryMatch (sm_svc) = "false" 144) => 145) Type_X_FindTagQuery.ind {}	146) Same
147) R13	148) Operational	149) RcvMsg() = "SM_FindTagQuery" 150) && FdaAddressType (sm_svc) = "HSE SMK" 151) && (QueryType (sm_svc) = “Function Block Tag” 152) || QueryType (sm_svc) = "Element ID") 153) => 154) SM_FindTagQuery.ind {}	155) Same
156) R14	157) Operational	158) RcvMsg() = "SM_FindTagQuery" 159) && FdaAddressType (sm_svc) = "GROUP SMK" 160) && (QueryType (sm_svc) = “Function Block Tag” 161) || QueryType (sm_svc) = "Element ID") 162) => 163) SM_FindTagQuery.ind {} 164) Type_X_FindTagQuery.ind {}	165) Same
166) R15	167) Operational	168) RcvMsg() = "SM_FindTagQuery" 169) && (FdaAddressType (sm_svc) = "Type_X SMK" 170) || FdaAddressType (sm_svc) = "LD Type_X INTERFACE SMK" 171) && (QueryType (sm_svc) = “Primary PD Tag” 172) || QueryType (sm_svc) = “Function Block Tag” 173) || QueryType (sm_svc) = "Element ID" 174) || QueryType (sm_svc) = "VFD Tag") 175) => 176) Type_X_FindTagQuery.ind {}	177) Same
178) R16	179) Operational	180) RcvMsg() = "SM_FindTagReply" 181) && FdaAddressType (sm_svc) = "HSE SMK" 182) && DuplicateQueryIdMatch (sm_svc) = "true" 183) && DevId_Match (sm_svc) = "false" 184) => 185) Set_DuplicatePdTagFlag () 186) SM_DeviceAnnunciation.req {} 187) Restart_HseRepeatTimer ()	188) Same
189) R17	190) Operational	191) RcvMsg() = "SM_FindTagReply" 192) && FdaAddressType (sm_svc) = "HSE SMK" 193) && DuplicateQueryIdMatch (sm_svc) = "true" 194) && DevId_Match (sm_svc) = “true” 195) => 196) // Do nothing – the response is from this device	197) Same
198) R18	199) Operational	200) RcvMsg() = "SM_FindTagReply" 201) && FdaAddressType (sm_svc) = "HSE SMK" 202) => 203) SM_FindTagReply.ind {} 204) 205) * Deliver indication to the AP that issued the Find Tag Query. How the SMKPM locates the requesting AP is not specified. There may be more than one way to do this. Error checking is performed on the response by the requesting AP.	206) Same
207) R19	208) No Tag	209) RcvMsg() = "SM_IdentifyReq" 210) && (FdaAddressType (sm_svc) = "Type_X SMK" 211) || FdaAddressType (sm_svc) = "LD Type_X INTERFACE SMK") 212) => 213) SM_ConfirmedService.err { 214) sm_service_type = SvcType (sm_svc),	219) Same

#	Current state	**Event** or **condition** **=> action**	Next state
		215) error_class = "service" 216) error_code = "object state conflict " 217) addl_code = 19 218) }	
220) R20	221) Operational	222) RcvMsg() = "SM_IdentifyReq" 223) && FdaAddressType (sm_svc) = "Type_X SMK" 224) && SmCacheEntry(sm_svc) = "false" 225) => 226) SM_ConfirmedService.err { 227) sm_service_type = SvcType (sm_svc), 228) error_class = "access" 229) error_code = "object invalidated" 230) addl_code = 20 231) }	232) Same
233) R21	234) Operational 235) No Tag	236) RcvMsg() = "SM_IdentifyReq" 237) => 238) SM_Identify.rsp {}	239) Same
240) R22	241) Operational 242) No Tag	243) (RcvMsg() = "SM_SetAssignmentInfoReq" 244) || RcvMsg() = "SM_ClearAssignmentInfoReq" 245) || RcvMsg() = "SM_ClearAddressReq") 246) && FdaAddressType (sm_svc) = "LD Type_X INTERFACE SMK" 247) => 248) SM_ConfirmedService.err { 249) sm_service_type = SvcType (sm_svc), 250) error_class = "access" 251) error_code = "object access denied" 252) addl_code = 22 253) }	254) Same
255) R23	256) Operational 257) No Tag	258) (RcvMsg() = "SM_SetAssignmentInfoReq" 259) || RcvMsg() = "SM_ClearAssignmentInfoReq" 260) || RcvMsg() = "SM_ClearAddressReq") 261) && DevId_Match (sm_svc) = “false” 262) => 263) SM_ConfirmedService.err { 264) sm_service_type = SvcType (sm_svc), 265) error_class = "service" 266) error_code = “key parameter mismatch” 267) addl_code = GetAddlCode () 268) }	269) Same
270) R24	271) No Tag	272) (RcvMsg() = "SM_SetAssignmentInfoReq" 273) || RcvMsg() = "SM_ClearAssignmentInfoReq" 274) || RcvMsg() = "SM_ClearAddressReq") 275) && FdaAddressType (sm_svc) = "Type_X SMK" 276) => 277) SM_ConfirmedService.err { 278) sm_service_type = SvcType (sm_svc), 279) error_class = "service" 280) error_code = “object state conflict” 281) addl_code = 24 282) }	283) Same
284) R25	285) Operational	286) (RcvMsg() = "SM_SetAssignmentInfoReq" 287) || RcvMsg() = "SM_ClearAssignmentInfoReq" 288) || RcvMsg() = "SM_ClearAddressReq") 289) && FdaAddressType (sm_svc) = "Type_X SMK" 290) && DeviceRedundancyState () = "secondary" 291) => 292) SM_ConfirmedService.err { 293) sm_service_type = SvcType (sm_svc), 294) error_class = "access" 295) error_code = “object access denied” 296) addl_code = 25 297) }	298) Same
299) R26	300) Operational	301) (RcvMsg() = "SM_SetAssignmentInfoReq" 302) || RcvMsg() = "SM_ClearAssignmentInfoReq" 303) || RcvMsg() = "SM_ClearAddressReq") 304) && PdTag_Match() = "false" 305) => 306) SM_ConfirmedService.err { 307) sm_service_type = SvcType (sm_svc), 308) error_class = "service" 309) error_code = “key parameter mismatch” 310) addl_code = GetAddlCode ()	312) Same

#	Current state	**Event** or **condition** **=> action**	Next state
		311) }	
313) R27	314) Operational	315) RcvMsg() = "SM_SetAssignmentInfoReq" 316) && FdaAddressType (sm_svc) = "Type_X SMK" 317) => 318) Type_X_SetAssignmentInfo.ind{}	319) Same
320) R28	321) Operational 322) No Tag	323) RcvMsg() = "SM_SetAssignmentInfoReq" 324) && PdTagDeviceIndex_Check(sm_svc) = "false" 325) => 326) SM_ConfirmedService.err { 327) sm_service_type = SvcType (sm_svc), 328) error_class = "service" 329) error_code = “parameter inconsistent” 330) addl_code = GetAddlCode () 331) }	332) Same
333) R29	334) No Tag	335) RcvMsg() = "SM_SetAssignmentInfoReq" 336) && FdaAddressType (sm_svc) = "HSE SMK" 337) && DeviceRedundancyState () = “Primary” 338) => 339) Set_Assignment_Data (sm_svc) 340) Clear_DuplicatePdTagFlag () 341) SM_SetAssignmentInfo.rsp {} 342) SM_DeviceAnnunciation.req {} 343) Restart_HseRepeatTimer () 344) SM_FindTagQuery.req {}	345) Operational
346) R30	347) No Tag	348) RcvMsg() = "SM_SetAssignmentInfoReq" 349) && FdaAddressType (sm_svc) = "HSE SMK" 350) && DeviceRedundancyState () = “Secondary” 351) => 352) Set_Assignment_Data (sm_svc) 353) Clear_DuplicatePdTagFlag () 354) SM_SetAssignmentInfo.rsp {} 355) SM_DeviceAnnunciation.req {} 356) Restart_HseRepeatTimer ()	357) Operational
358) R31	359) Operational	360) RcvMsg() = "SM_SetAssignmentInfoReq" 361) && FdaAddressType (sm_svc) = "HSE SMK" 362) => 363) Set_Assignment_Data (sm_svc) 364) SM_SetAssignmentInfo.rsp {} 365) SM_DeviceAnnunciation.req {} 366) Restart_HseRepeatTimer ()	367) Same
368) R32	369) No Tag	370) RcvMsg() = "SM_ClearAssignmentInfoReq" 371) && FdaAddressType (sm_svc) = "HSE SMK" 372) => 373) // This will force the device back to defaults 374) RestoreDefaults () 375) SM_ClearAssignmentInfo.rsp {} 376) SM_DeviceAnnunciation.req {} 377) Restart_HseRepeatTimer ()	378) Same
379) R33	380) Operational	381) RcvMsg() = "SM_ClearAssignmentInfoReq" 382) && FdaAddressType (sm_svc) = "Type_X SMK" 383) => 384) Type_X_ClearAssignmentInfo.ind{}	385) Same
386) R34	387) Operational	388) RcvMsg() = "SM_ClearAssignmentInfoReq" 389) && FdaAddressType (sm_svc) = "HSE SMK" 390) => 391) RestoreDefaults () 392) SM_ClearAssignmentInfo.rsp {} 393) SM_DeviceAnnunciation.req {} 394) Restart_HseRepeatTimer ()	395) No Tag
396) R35	397) Operational	398) RcvMsg() = "SM_ClearAddressReq" 399) && FdaAddressType (sm_svc) = "Type_X SMK" 400) => 401) Type_X_ClearAddress.ind{}	402) Same
403) R36	404) Operational 405) No Tag	406) RcvMsg() = "SM_ClearAddressReq" 407) && FdaAddressType (sm_svc) = "HSE SMK" 408) && (AddressToClear(sm_svc) = "None" 409) || ConfigurationSessionActive = “True”) 410) => 411) SM_ConfirmedService.err { 412) sm_service_type = SvcType (sm_svc), 413) error_class = "service" 414) error_code = “object constraint	417) Same

#	Current state	**Event** or **condition** **=> action**	Next state
		conflict” 415) addl_code = GetAddlCode () 416) }	
418) R37	419) Operational 420) No Tag	421) RcvMsg() = "SM_ClearAddressReq" 422) && FdaAddressType (sm_svc) = "HSE SMK" 423) && (NumberOfAssignedAddresses() = 1 424) || AddressToClear(sm_svc) = "All") 425) => 426) SM_ClearAddress.rsp {} 427) Clear_Address (interface_to_clear)	428) No Address
429) R38	430) Operational 431) No Tag	432) RcvMsg() = "SM_ClearAddressReq" 433) && FdaAddressType (sm_svc) = "HSE SMK" 434) && NumberOfAssignedAddresses() > 1 435) => 436) Clear_Address (interface_to_clear) 437) SM_ClearAddress.rsp {} 438) SM_DeviceAnnunciation.req {} 439) Restart_HseRepeatTimer ()	440) Same

Table 95 – SMKPM state table – internal events

#	Current state	Event or condition => action	Next state
441) S1	442) Operational 443) No Tag	444) HseRepeatTimerExpires() 445) => 446) SM_DeviceAnnunciation.req {} 447) Restart_HseRepeatTimer ()	448) Same
449) S2	450) No Address	451) RcvNewNetworkAddress (interface, address) 452) && AssignmentInfo_Set () = "false" 453) => 454) RestoreDefaults () 455) NewAddress (interface, address) 456) SM_DeviceAnnunciation.req {} 457) Restart_HseRepeatTimer ()	458) No Tag
459) S3	460) No Address	461) RcvNewNetworkAddress (interface, address) 462) && AssignmentInfo_Set () = "true" 463) && DeviceRedundancyState () = “Primary” 464) => 465) Clear_DuplicatePdTagFlag () 466) NewAddress (interface, address) 467) SM_DeviceAnnunciation.req {} 468) Restart_HseRepeatTimer () 469) SM_FindTagQuery.req {}	470) Operational
471) S4	472) No Address	473) RcvNewNetworkAddress (interface, address) 474) && AssignmentInfo_Set () = "true" 475) && DeviceRedundancyState () = “Secondary” 476) => 477) Clear_DuplicatePdTagFlag () 478) NewAddress (interface, address) 479) SM_DeviceAnnunciation.req {} 480) Restart_HseRepeatTimer ()	481) Operational

482)
S5
	483) Operational	484) // for new address for second HSE interface or for changing the address of either
		485) // HSE interface
		486) RcvNewNetworkAddress (interface, address)
		487) && NetworkAddressChange (interface, address) = “false”
		488) =>
		489) NewAddress (interface, address)
		490) SM_DeviceAnnunciation.req {}
		491) Restart_HseRepeatTimer ()
			492) Same

493) S6	494) Operational	495) SM_UnconfirmedService.req {} 496) || SM_ConfirmedService.req {} 497) || SM_ConfirmedService.rsp {} 498) || Type_X_ConfirmedService.rsp {} 499) => 500) Send_SM_ReqRspMessage (sm_svc)	501) Same
502) S7	503) Operational	504) SM_ConfirmedService.err{} 505) || Type_X_ConfirmedService.err {} 506) => 507) Send_SM_CommonErrorRsp (508) sm_service_type = SvcType (sm_svc), 509) error_class = "result error class" 510) error_code = “result error code” 511) addl_code = “result error addl_code” 512) addl_description = “result error addl_description”) 513))	514) Same
515) S8	516) Operational 517) No Tag	518) SntpSyncLost() 519) => 520) SM_DeviceAnnunciation.req {} 521) Restart_HseRepeatTimer ()	522) Same

4.5.7 Function descriptions

4.5.7.1 General

The following descriptions define the functions referenced by the preceding state tables.

4.5.7.2 Event functions

4.5.7.2.1 Summary

These functions are *callback* functions used to receive an event. They do not cause any modification of the data received. The other events are Service Primitives defined previously. See Table 96 through Table 99 for definitions of the functions.

4.5.7.2.2 HseRepeatTimerExpires ()

Table 96 – HseRepeatTimerExpires ()

Function	
This function is invoked when the Hse Repeat Timer expires.	
Input parameters	Output parameters
none	none

4.5.7.2.3 RcvNewNetworkAddress (interface, address)

Table 97 – RcvNewNetworkAddress (interface, address)

Function	
This function is invoked when an network address is received from the underlying layers. The input parameters identify the Type 5 interface and the received network address.	
Input parameters	Output parameters
interface, address	none

4.5.7.2.4 RcvMsg ()

Table 98 – RcvMsg ()

Function	
This function is invoked when an APDU is received. It conveys the sm_svc parameter to the state machine. This parameter contains all the fields of the APDU, including the header and trailer fields. The syntax RcvMsg() = "SM_XXX" is used to indicate that an APDU for SM service XXX has been received.	
Input parameters	Output parameters
sm_svc	none

4.5.7.2.5 SntpSyncLost ()

Table 99 – SntpSyncLost ()

Function	
This function is invoked when the state of synchronization of time between the device and the selected remote time server changes from true to false.	
Input parameters	Output parameters
none	none

4.5.7.3 Condition functions

4.5.7.3.1 Summary

These functions are used to test an event. They do not cause any modification of the data tested. See Table 100 through Table 116 for definitions of the functions.

4.5.7.3.2 AddressToClear (sm_svc)

Table 100 – AddressToClear (sm_svc)

Function	
Returns which Type 5 interfaces are being requested to be cleared. The values are OperationalNetworkAddress, Non-OperationalNetworkAddress, All, and None. None indicates that the specified Type 5 interface does not have an address or is not capable of dynamically requesting a new address.	
Input parameters	Output parameters
none	HseInterface

4.5.7.3.3 AssignmentInfo_Set ()

Table 101 – AssignmentInfo_Set ()

Function	
Returns true if the SMK assignment information has been previously set by the Set Assignment service and is currently still valid as defined by that service.	
Input parameters	Output parameters
none	true or false

4.5.7.3.4 ConfigurationSessionActive ()

Table 102 – ConfigurationSessionActive ()

Function	
Returns true if there is a configuration AR established.	
Input parameters	Output parameters
none	true or false

4.5.7.3.5 DeviceRedundancyState ()

Table 103 – DeviceRedundancyState ()

Function	
Returns the redundancy state of the device.	
Input parameters	Output parameters
none	primary or secondary

4.5.7.3.6 DevId_Match (sm_svc)

Table 104 – DevId_Match (sm_svc)

Function	
Returns true if the Device ID in the APDU exactly matches the Device ID of this device.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.3.7 DuplicateQueryIdMatch (sm_svc)

Table 105 – DuplicateQueryIdMatch (sm_svc)

Function	
Returns true if the Invoke Id matches the Invoke Id of the Find Tag Request that was sent to determine if another device has the same PD Tag as this device.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.3.8 DuplicatePdTagDetected ()

Table 106 – DuplicatePdTagDetected ()

Function	
Returns true if the device has detected another device with the same Physical Device Tag as this device.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.3.9 FdaAddressType (sm_svc)

Table 107 – FdaAddressType (sm_svc)

Function	
Evaluates the FDA Address component of the sm_svc parameter and returns which SMK it identifies. For the return values defined below, LL LL identifies an Type "X" link connected to the linking device, and NN is the node address for that link interface. Returns: Type 5 SMK if FDA Address = 00 00.00.SM, where SM is the SMK selector GROUP SMK if FDA Address = 00 00.GG GG or LL LL.GG GG, where GG GG is the Type "X" SMK group address LD Type "X" Interface SMK if FDA Address = LL LL.NN.SM and NN is the node address of an Type "X" interface of the LD, Type "X" SMK if FDA Address = LL LL.NN.SM UNRECOGNIZED if FDA Address is not one of the above	
Input parameters	Output parameters
sm_svc	SMK Identifier

4.5.7.3.10 IsValid (sm_svc)

Table 108 – IsValid (sm_svc)

Function	
Returns true if the parameters in the APDU are valid for the service type. Data is considered valid if it has the correct data type and it conforms to the values defined for it. Parameter values are defined in the SMK service specifications in IEC 61158-5 and in the APDU specifications in this specification. The values are checked for validity in this function. Note the contrast with other condition functions defined for the SMKPM, such as PdTag_Match(sm_svc) that check for the use of the value.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.3.11 NetworkAddressChange (interface, address)

Table 109 – NetworkAddressChange (interface, address)

Function	
Returns true if the specified interface already has a valid network address.	
Input parameters	Output parameters
interface, address	true or false

4.5.7.3.12 NumberOfAssignedAddresses ()

Table 110 – NumberOfAssignedAddresses ()

Function	
Returns the number of Type 5 Interfaces that currently have an network address.	
Input parameters	Output parameters
none	NumberOfInterfaces

4.5.7.3.13 OperationalRestore ()

Table 111 – OperationalRestore ()

Function	
Returns true if Operational Powerup is true and the last state of the SMKPM was Operational.	
Input parameters	Output parameters
none	true or false

4.5.7.3.14 PdTag_Match (sm_svc)

Table 112 – PdTag_Match (sm_svc)

Function	
Returns true if the PD Tag in the APDU exactly matches the PD Tag of the device identified by the FDA Address of the APDU. Group FDA Addresses cause false to be returned.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.3.15 PdTagDeviceIndex_Check (sm_svc)

Table 113 – PdTagDeviceIndex_Check (sm_svc)

Function	
Returns false if: 1. the service would cause PD Tag or the Device Index to be cleared or remain clear (both shall be set), or. 2. the PD Tag in the device is set before the service begins and the service is attempting to change it (the device index can be changed, but the PD Tag cannot). Otherwise, returns true.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.3.16 Query_Match (sm_svc)

Table 114 – Query_Match (sm_svc)

Function	
Returns true if the device contains the queried object.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.3.17 QueryType (sm_svc)

Table 115 – QueryType (sm_svc)

Function	
Returns the Query Type for a Find Tag Query or Find Tag Reply.	
Input parameters	Output parameters
sm_svc	query type

4.5.7.3.18 SmCacheEntry (sm_svc)

Table 116 – SmCacheEntry (sm_svc)

Function	
Returns true if the SMK has cached the identification information for the Type "X" device identified by the fda_address component of the sm_svc parameter.	
Input parameters	Output parameters
sm_svc	true or false

4.5.7.4 Action functions

4.5.7.4.1 Summary

These functions are used in the Action portion of the transitions. See Table 117 through Table 129 for definitions of the functions.

4.5.7.4.2 Clear_Address (interface_to_clear)

Table 117 – Clear_Address (interface_to_clear)

Function	
Close all FDA sessions and Type 9 VCRs for the address to be cleared. Stop the Type 5 Repeat Timer if it is running. Clear the address for the specified Type 5 interface in the local network address array . Request the underlying layers to clear the address. If the Operational network address is no longer valid, set it to the address of the other Type 5 interface if that address is valid.	
Input parameters	Output parameters
interface_to_clear	none

4.5.7.4.3 Clear_DuplicatePdTagFlag ()

Table 118 – Clear_DuplicatePdTagFlag ()

Function	
Performs the equivalent of DuplicateDetectedState := DuplicateDetectedState & NOT “DuplicatePdTag”	
Input parameters	Output parameters
none	none

4.5.7.4.4 Get_AddlCode ()

Table 119 – Get_AddlCode ()

Function	
Determines the value of the additional code variable. These values are defined in 4.5.6.	
Input parameters	Output parameters
none	addl_code

4.5.7.4.5 New_Address (interface, address)

Table 120 – New_Address (interface, address)

Function	
If the existing network address for the specified Type 5 interface is valid, close all client/server FDA sessions and Type 9 VCRs for that address. Set or change the address for the specified Type 5 interface in the local network address array. If the Operational network address is not valid or no longer valid, set it to this address. If the Operational Network Address was valid and was changed, then immediately move all existing publisher and report source VCRs and FDA Sessions to the new Operational Network Address.	
Input parameters	Output parameters
interface, address	none

4.5.7.4.6 Restart_HseRepeatTimer ()

Table 121 – Restart_HseRepeatTimer ()

Function	
This function stops the HseRepeatTimer if it is running, and restarts it with the period of time defined by the Type 5 Repeat Time attribute. When this timer expires, it generates the HseRepeatTimerExpires event for the SM state machine.	
Input parameters	Output parameters
None	None

4.5.7.4.7 Restore_Defaults ()

Table 122 – Restore_Defaults ()

Function	
Restore this device to its 'as manufactured' condition, except that network addresses and the Operational network address are retained. This closes all sessions and VCRs and removes all configured and operational data.	
Input parameters	Output parameters
none	none

4.5.7.4.8 Send_SM_CommonErrorRsp (sm_service_type, svc_spec_params)

Table 123 – Send_SM_CommonErrorRsp (sm_service_type, svc_spec_params)

Function	
Builds and sends the error response APDU for the service identified in the sm_service_type parameter.	
Input parameters	Output parameters
sm_service_type, svc_spec_params	none

4.5.7.4.9 Send_SM_ReqRspMessage (sm_svc)

Table 124 – Send_SM_ReqRspMessage (sm_svc)

Function	
Builds and sends the request or response APDU for the service identified in the sm_svc parameter.	
Input parameters	Output parameters
sm_svc	none

4.5.7.4.10 Set_Assignment_Data (sm_svc)

Table 125 – Set_Assignment_Data (sm_svc)

Function	
Copies assignment data for an Type 5 device from the APDU to matching SMK attributes. If the Operational IP Address changes, then immediately move all existing publisher and report source VCRs and FDA Sessions to the Operational IP Address.	
Input parameters	Output parameters
sm_svc	none

4.5.7.4.11 Set_DuplicatePdTagFlag ()

Table 126 – Set_DuplicatePdTagFlag ()

Function	
Sets the DuplicateDetectedState attribute to reflect that this device has detected that this device and another device have the same PD Tag.	
Input parameters	Output parameters
none	none

4.5.7.4.12 SvcType (sm_svc)

Table 127 – SvcType (sm_svc)

Function	
This function decodes the data that is conveyed in the sm_svc parameter and returns the service type.	
Input parameters	Output parameters
sm_svc	service type

4.5.8 Error classes and codes

The Error Classes and Codes defined in 4.3.6.4 apply.

4.5.9 Additional code definition

Table 697 and Table 698 define the values that may be used in the Additional Code octet of the FDA Common Error classes. This definition is local to SM services.

Table 128 – Additional code used by error class and code

FDA error class	FDA error code	Additional code value for SMKPM	Provided By
service	key parameter mismatch	see Table 129	GetAddlCode ()
	parameter inconsistent	see Table 129	GetAddlCode ()
	object constraint conflict	see Table 129	GetAddlCode ()

Table 129 – Additional code parameter IDs

Parameter	Additional code
Clear Duplicate Detection State	5
Device ID	10
Device Index	20
Device Redundancy State	25
Element ID	30
FDA Address	35
Function Block Tag	40
Type "X" New Address	45
Interface to Clear	50
Invoke ID	55
PD Tag	60
Type 5 Repeat Time	65

Parameter	Additional code
LAN Redundancy Addr	70
Max Device Index	75
Operational IP Address	80
Query Type	85
VFD Reference	90
VFD Tag	95
Interface does not have an address	101
Interface has a fixed address	102
Interface has a configuration session	103

4.6 VCR state machine

The Type 5 VCR state machine is defined by the Type 9 VCR State Machine, with the following modifications.

- The Type 5 Initiate service replaces the Type 9 Initiate service. Type 9 Initiate service parameters that are not defined for the Type 5 Initiate service are ignored in the state machine.
- References to the AR Abort service by the Type 9 Connection Establishment state transitions are modified as follows. The Abort request primitive is replaced by the sending of an Type 5 VCR Abort request APDU, and the receipt of a Type 9 Abort indication primitive is replaced by the receipt of an Type 5 VCR Abort request APDU. The only exception to this rule is when an Type 9 Abort.request primitive is issued in response to the receipt of an Type 9 Initiate request PDU when the VCR is in the Connection Established state. In this case, the FDA Agent returns an Type 5 Initiate Error APDU with the same error class and error code supplied in the Type 9 Abort.request primitive.
- The Type 5 VCR Client and Server endpoints use Type 5 Idle APDUs as "keep alive" APDUs as follows.
- Client Type 5 VCR endpoints send Idle request APDUs only when the Idle timer expires.
- Server Type 5 VCR endpoints immediately send Idle Response APDUs when they receive Idle request APDUs.
- Client endpoints restart their Idle Timers whenever they send a Type 5 APDU.
- Server endpoints do not maintain an Idle timer.
- Upon expiration of their Inactivity timer, client and server Type 5 VCR endpoints terminate after sending an Abort APDU.
- Requests to establish an Type 5 VCR of type "Type 9 CLIENT" using the VCR Selector option are rejected if the remote selector address of the Type 9 Client VCR indicates that the remote VFD is the MIB VFD. The error class used is "access" and the error code is "object access denied".
- Server endpoints established for MIB access respond negatively to received Type 9 update service requests using error class "access" and error code "object access denied" if the underlying AR is not a configuration AR. The Type 9 update service requests are:
  - 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

4.7 FAL service protocol machine (FSPM)

4.7.1 Type 5 FSPM

The formal interface between an Type 5 VCR and its underlying AR maintains compatibility with the Type 9 VCR state transitions used for the Type 5 VCR State Machine.

The Type 5 FSPM is defined by the Type 9 FSPM. It is used in its entirety with the following modifications:

- The Confirmed Send and Unconfirmed Send service primitives for sending and receiving have been extended to contain the Type 5_VCR_ID parameter to identify the VCR that uses the session endpoint.
- All references to remote DLCEP addresses are replaced by references to remote addresses. When used with ARs that operate over connections, the remote address parameter identifies the connection.
- The Abort service is not conveyed between the FSPM and the Type 5 VCR state machines. Instead, the Type 5 VCR Abort APDU is conveyed between the two state machines using the Unconfirmed Send service.
- Some service parameter differences exist in the service primitives exchanged between the Type 5 FSPM and the Type 5 ARPM. The service primitives and their parameters are defined in 4.8.1.1.1.

4.8 Application relationship protocol machines (ARPMs)

4.8.1 Client / server session ARPM

4.8.1.1 Primitive definitions

4.8.1.1.1 Primitives exchanged between ARPM and FSPM

Table 130 and Table 131 define the primitives exchanged between the ARPM and the FSPM.

Table 130 – Primitives issued by FSPM to ARPM

Primitive name	Source	Associated parameters	Functions
EST_req	FSPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish request primitive from the FSPM to the ARPM.
EST_rsp(+)	FSPM	arep_id, user_data,	This is an FAL internal primitive used to convey an Establish response(+) primitive from the FSPM to the ARPM.
EST_rsp(-)	FSPM	arep_id, user_data, service_type	This is an FAL internal primitive used to convey an Establish response(-) primitive from the FSPM to the ARPM.
Abort_req	FSPM	vcr_id, abort_detail, abort_identifier, reason_code	This is an FAL internal primitive used to convey an Abort request primitive from the FSPM to the ARPM.
CS_req	FSPM	vcr_id, service_type, user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) request primitive from the FSPM to the ARPM.
CS_rsp	FSPM	vcr_id, service_type, user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) response primitive from the FSPM to the ARPM.
523) UCS_req	524) FS PM	525) vcr_id, 526) service_type, 527) user_data	528) This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) request primitive from the FSPM to the ARPM.

Table 131 – Primitives issued by ARPM to FSPM

Primitive name	Source	Associated parameters	Functions
EST_ind	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish indication primitive from the ARPM to the FSPM.
EST_cnf(+)	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish confirmation(+) primitive from the ARPM to the FSPM.
EST_cnf(-)	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish confirmation(-) primitive from the ARPM to the FSPM.
Abort_ind	ARPM	vcr_id, abort_detail, abort_identifier, reason_code	This is an FAL internal primitive used to convey an Abort primitive from the ARPM to the FSPM.
CS_ind	ARPM	vcr_id, service_type, user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) indication primitive from the ARPM to the FSPM.
CS_cnf(+)	ARPM	vcr_id, service_type, user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) confirmation(+) primitive from the ARPM to the FSPM.
CS_cnf(-)	ARPM	vcr_id, service_type, user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) confirmation(-) primitive from the ARPM to the FSPM.
529) UCS_ind	530) A RPM	531) vcr_id, 532) service_type, 533) user_data	534) This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) indication primitive from the ARPM to the FSPM.

4.8.1.1.2 Parameters of FSPM/ARPM primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in Table 132.

Table 132 – Parameters used with primitives exchanged between FSPM and ARPM

Parameter name	Description
arep_id	This parameter is used to unambiguously identify an instance of the AREP that has issued a primitive. A means for such identification is not specified by this standard.
user_data	This parameter conveys FAL-User data.
abort_identifier	This parameter conveys the value that is used for the Abort_Identifier parameter.
reason_code	This parameter conveys the value that is used for the Reason_Code parameter.
abort_detail	This parameter conveys the value that is used for the Abort_Detail parameter.
vcr_id	This parameter is used to unambiguously identify an instance of the VCR that issued a primitive or that is the primitive destination. A means for such identification is not specified by this standard.
535) service_type	536) This parameter conveys the protocol Id and the service within the protocol.

4.8.1.2 DLL mapping of Client / Server AREP Class

4.8.1.2.1 Formal Definition

Subclause 4.8.1.2 describes the mapping of the Client / Server AREP Class to the Socket model. The DLL mapping attributes and their permitted values used with the Client / Server AREP class are defined in 4.8.1.2.

CLASS: **ClientServer**
PARENT CLASS: **AR Endpoint**
ATTRIBUTES:

1	(m)	KeyAttribute:	LocalAndRemoteAddresses
1.1	(m)	KeyAttribute:	LocalAddress
1.2	(m)	KeyAttribute:	RemoteAddress
2	(m)	Attribute:	TransferType (CONN, CNLS)
3	(m)	Attribute	BufferSize
4	(m)	Attribute:	TransmitDelayTime
5	(m)	Attribute:	InactivityCloseTime
6	(m)	Attribute:	HdrOptions
7	(m)	Attribute:	GuardBand
8	(m)	Attribute:	MibConfigurationUse
9	(m)	Attribute:	MaxApduLength
10	(m)	Attribute:	MaxOutstanding
11	(m)	Attribute:	ServerPhysicalDeviceTag

4.8.1.2.2 Attributes

LocalAndRemoteAddresses

This key attribute identifies the AREP. It is composed of the local and remote addresses of the underlying layer for the AREP.

TransferType

This attribute specifies whether the connection-oriented (CONN) or connectionless (CNLS) services of the underlying layer are used by the AREP.

BufferSize

This attribute specifies the size in octets of the session endpoint's send buffer. The buffer holds an integral number of APDUs. If an APDU is received for transfer that would overflow the buffer, the buffer is sent first, and the APDU is then added as the first APDU in the buffer.

TransmitDelayTime

This attribute specifies the amount of time that a sending session endpoint accumulates APDUs to send in its buffer. The time period starts when the first APDU is placed into the transmit buffer after either the buffer has been initialized or after it has been emptied by the previous transmission.

If the endpoint has no APDUs to send (the buffer is empty) and it receives an APDU to send, it loads the APDU into the buffer and waits for additional APDUs to concatenate into the buffer. If the buffer fills before the Transmit Delay Time has been reached, the Type 5 FAL AE stops the timer and sends the buffer. When this time interval has expired and the buffer has not yet filled, it sends the contents of the buffer.

InactivityCloseTime

This attribute is used by Client and Server endpoints to determine when to close their endpoints because of inactivity.

HdrOptions

This attribute specifies which APDU Trailer fields are present in the APDUs sent and received on the session.

GuardBand

This attribute specifies the lower and upper bound for the field being protected from rollover.

MibConfigurationUse

This Boolean attribute specifies, when TRUE, that the AR is used for MIB configuration. This is not a configurable attribute. Client applications are expected to know that they need configuration ARs and request them with the Establish request.

MaxApduLength

This attribute specifies the maximum permitted length in octets of APDUs sent by the CLIENT AREPs and received by SERVER AREPs.

MaxOutstanding

This attribute specifies the maximum number of requests that the receiving endpoint may have outstanding at any point in time. Outstanding means that the endpoint has received a request message, but has not returned the corresponding response message. Its initial value is set by the device manufacturer. If a request message is received that causes this attribute to be exceeded, a negative response is returned with error class = "resource" and error code = "max outstanding requests per session exceeded".

ServerPhysicalDeviceTag

This attribute specifies the SMK PhysicalDeviceTag of the server endpoint.

4.8.1.3 Client / Server AREP state machine

4.8.1.3.1 Client / Server ARPM states

The defined states and their descriptions of the Client / Server ARPM are shown in Table 133 and Figure 2. If the transfer type = CONN, then the AR operates over an underlying connection, and the state machine does not take effect until the underlying connection has been established. If the first message received on the underlying connection is not an Establish request, then the underlying connection is terminated.

Table 133 – Client / Server ARPM states

States	Description
CLOSED	Client and Server AREPs shall be created prior to use. They are created in the CLOSED state. When the FDA Agent receives an OpenSession Request FDA-PDU it dynamically creates a new server AREP and delivers the received FDA-PDU to it. AREPs in the CLOSED state are not capable of sending or receiving FDA-PDUs except for OpenSession FDA-PDUs. Transitioning an AREP to the CLOSED state causes its timers to be stopped, its underlying socket connection to be closed if necessary, and it is then deleted.
OPEN	The AREP is defined and capable of sending or receiving FAL-PDUs.
REQUESTING	The AREP has sent an Establish Request FAL-PDU and is waiting for a response from the remote AREP.
RESPONDING	The AREP has received an Establish Request FAL-PDU, delivered an Establish.ind primitive and is waiting for a response from its user.
537) SAME	538) Indicates that the next state is the same as the current state.

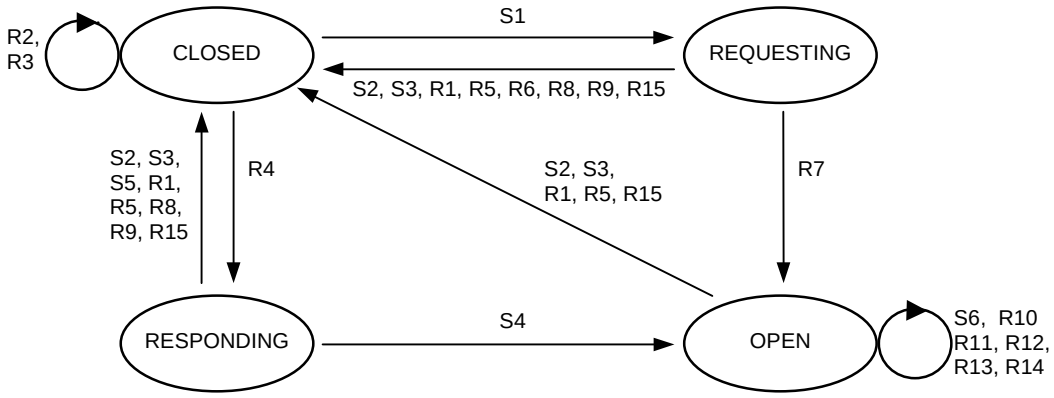


Figure 2 – State transition diagram of client / server ARPM

4.8.1.3.2 Client / server ARPM state table

Table 134 and Table 135 define the Client/Sever ARPM state machine.

Table 134 – Client / server ARPM state table – sender transitions

#	Current state	Event or condition => action	Next state
539) S1	540) CLOSED	541) EST_req 542) && EndpointType = "CLIENT" 543) => 544) FAL-PDU_req { 545) smpm_service_name = “SMPM_Send”, 546) arep_id = GetArepId (), remote_address = RemoteAddress, 547) fda_pdu = BuildFAL-ReqRspPDU (service_type = “AR Establish Req”, 548) fal_data = user_data) 549) }, 550) 551) StartInactivityCloseTimer()	552) REQUESTING
553) S2	554) RESPONDING 555) REQUESTING 556) OPEN	557) Abort_req 558) =>	559) CLOSED
560) S3	561) REQUESTING RESPONDING 562) OPEN	563) InactivityCloseTimerExpires 564) =>	565) CLOSED
566) S4	567) RESPONDING	568) EST_rsp(+) 569) => 570) FAL-PDU_req { 571) smpm_service_name = “SMPM_Send”, 572) arep_id = GetArepId (), remote_address = RemoteAddress, 573) fda_pdu = BuildFAL-ReqRspPDU (service_type = “AR Establish Rsp”, 574) fal_data = user_data) 575) } 576) 577) StartInactivityCloseTimer()	578) OPEN
579) S5	580) RESPONDING	581) EST_rsp(-) 582) => 583) FAL-PDU_req { 584) smpm_service_name = “SMPM_Send”, 585) arep_id = GetArepId (), remote_address = RemoteAddress, 586) fda_pdu = BuildFAL-ReqRspPDU (service_type = “AR Establish Err”, 587) fal_data = user_data) 588) }	589) CLOSED
590) S6	591) OPEN	592) UCS_req 593) || CS_req 594) || CS_rsp 595) => 596) FAL-PDU_req { 597) smpm_service_name = “SMPM_Send”, 598) arep_id = GetArepId (), remote_address = RemoteAddress, 599) fda_pdu = BuildFAL-ReqRspPDU (service_type = service_type, 600) fal_data = user_data) 601) }	602) OPEN

Table 135 – Client / server ARPM state table – receiver transitions

#	Current state	Event or condition => action	Next state
603) R1	604) REQUESTING 605) RESPONDING OPEN	606) FAL-PDU_ind 607) && FDA_PDU_Valid(fal_pdu) = False 608) => 609) FAL-PDU_req* { 610) smpm_service_name = “SMPM_Send”, 611) arep_id = GetArepId (), 612) remote_address = FAL_Pdu_RemoteAddress(fal_pdu), 613) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 614) error_class = “service”, 615) error_code = appropriate error code for detected error) 616) } 617) 618) * Used only when the received FAL PDU is a DTC Request	619) CLOSED
620) R2	621) CLOSED	622) FAL-PDU_ind 623) && EndpointType = “SERVER” 624) && FAL_Pdu_SvcType (fda_pdu) = “AR Establish Req” 625) && FAL_PDU_Valid(fda_pdu) = False 626) => 627) FAL-PDU_req { 628) smpm_service_name = “SMPM_Send”, 629) arep_id = GetArepId (), 630) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 631) fda_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 632) error_class = “service”, 633) error_code = appropriate error code for detected error) 634) }	635) CLOSED
636) R3	637) CLOSED	638) FAL-PDU_ind 639) && EndpointType = “SERVER” 640) && FAL_Pdu_SvcType (fal_pdu) = “AR Establish Req” 641) && ConfigurationArCheckOK (fal_pdu) = False 642) => 643) FAL-PDU_req { 644) smpm_service_name = “SMPM_Send”, 645) arep_id = GetArepId (), 646) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 647) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 648) error_class = “access”, 649) error_code = “config access already open”) 650) }	651) CLOSED
652) R4	653) CLOSED	654) FAL-PDU_ind 655) && EndpointType = “SERVER” 656) && FAL_Pdu_SvcType (fal_pdu) = “AR Establish Req” 657) && ConfigurationArCheckOK(fal_pdu) = True 658) => 659) RemoteAddress = FAL_Pdu_RemoteAddress () 660) EST_ind* { 661) arep_id = GetArepId (), 662) user_data = GetUserData (fal_pdu) 663) } 664) StartInactivityCloseTimer() 665) 666) * The server entity that receives this indication is responsible for performing parameter negotiations.	667) RESPON DING
668) R5	669) OPEN 670) RESPONDING 671) REQUESTING	672) FAL-PDU_ind 673) && FAL_Pdu_SvcType (fal_pdu) = “AR Establish Req” 674) => 675) FAL-PDU_req { 676) smpm_service_name = “SMPM_Send”, 677) arep_id = GetArepId (), 678) remote_address = FAL_Pdu_RemoteAddress (), 679) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 680) error_class = “service”, 681) error_code = "object state conflict") 682) }	683) CLOSED
684)	685) REQUESTING	686) FAL-PDU_ind	693) CLOSED

#	Current state	Event or condition => action	Next state
R6		687) && FAL_Pdu_SvcType (fal_pdu) = “AR Establish Err” 688) => 689) EST_cnf(-) { 690) arep_id = GetArepId (), 691) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 692) }	
694) R7	695) REQUESTING	696) FAL-PDU_ind 697) && FAL_Pdu_SvcType (fal_pdu) = “AR Establish Rsp” 698) => 699) BufferSize = FAL_Pdu_BufferSize(fal_pdu) 700) InactivityCloseTime = GetInactivityCloseTime(fal_pdu) 701) EST_cnf(+) { 702) arep_id = GetArepId (), 703) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 704) } 705) StartInactivityCloseTimer()	706) OPEN
707) R8	708) RESPONDING 709) REQUESTING	710) FDA-PDU_ind 711) && FDA_Pdu_SvcType (fda_pdu) = “CS_ReqPDU” 712) => 713) FDA-PDU_req { 714) smpm_service_name = “SMPM_Send”, 715) arep_id = GetArepId (), 716) remote_address = FDA_Pdu_RemoteAddress (fda_pdu), 717) fda_pdu = BuildFDA-ErrPDU(request_pdu = fda_pdu, 718) error_class = “service”, 719) error_code = “object state conflict”) 720) } 721) StartInactivityCloseTimer()	722) CLOSED
723) R9	724) RESPONDING	725) FAL-PDU_ind 726) && FAL_Pdu_SvcType (fal_pdu) = “ANY FAL PDU” 727) =>	728) CLOSED
729) R10	730) OPEN	731) FAL-PDU_ind 732) && FAL_Pdu_SvcType (fal_pdu) = “UCS_ReqPDU” 733) && FAL_Pdu_GetVcrId() <> NULL 734) => 735) UCS_ind { 736) arep_id = GetArepId (), 737) vcr_id = FAL_Pdu_GetVcrId(), 738) service_type = FAL_Pdu_SvcType(fal_pdu), 739) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 740) } 741) StartInactivityCloseTimer()	742) OPEN
743) R11	744) OPEN	745) FAL-PDU_ind 746) && FAL_Pdu_SvcType (fal_pdu) = “CS_ReqPDU” 747) && FAL_Pdu_GetVcrId() = NULL 748) => 749) FAL-PDU_req { 750) smpm_service_name = “SMPM_Send”, 751) arep_id = GetArepId (), 752) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 753) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 754) error_class = “access”, 755) error_code = “unrecognized FDA Address”) 756) } 757) StartInactivityCloseTimer()	758) OPEN
759) R12	760) OPEN	761) FAL-PDU_ind 762) && FAL_Pdu_SvcType (fal_pdu) = “CS_ReqPDU” 763) => 764) CS_ind { 765) arep_id = GetArepId (), 766) vcr_id = FAL_Pdu_GetVcrId(), 767) service_type = FAL_Pdu_SvcType(fal_pdu), 768) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 769) } 770) StartInactivityCloseTimer()	771) OPEN
772) R13	773) OPEN	774) FAL-PDU_ind 775) && FAL_Pdu_SvcType (fda_pdu) = “DTC_ReqPDU” 776) && MaxOutstandingReached() = True 777) => 778) FAL-PDU_req { 779) smpm_service_name = “SMPM_Send”, 780) arep_id = GetArepId (),	787) OPEN

#	Current state	Event or condition => action	Next state
		781) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 782) fda_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 783) error_class = “resource”, 784) error_code = “max outstanding requests per session exceeded”) 785) } 786) StartInactivityCloseTimer()	
788) R14	789) OPEN	790) FAL-PDU_ind 791) && (FAL_Pdu_SvcType (fal_pdu) = “CS_RspPDU” 792) || FAL_Pdu_SvcType (fal_pdu) = “CS_ErrPDU) 793) => 794) CS_cnf { 795) arep_id = GetArepId (), 796) vcr_id = FAL_Pdu_GetVcrId(), 797) service_type = FAL_Pdu_SvcType(fal_pdu), 798) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 799) } 800) StartInactivityCloseTimer()	801) OPEN
802) R15	803) OPEN REQUESTING 804) RESPONDING	805) ErrorToARPM 806) && ErrorType = “socket disconnect” 807) =>	808) CLOSED

4.8.2 ARPM

4.8.2.1 General

This ARPM is used by Publisher/Subscriber and Report Distribution ARs.

4.8.2.2 Primitive definitions

4.8.2.2.1 Primitives exchanged between ARPM and FSPM

Table 136 and Table 137 list the primitives exchanged between the ARPM and the FSPM.

Table 136 – Primitives issued by FSPM to ARPM

Primitive name	Source	Associated parameters	Functions
EST_req	FSPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish request primitive from the FSPM to the ARPM.
Abort_req	FSPM	vcr_id, abort_detail, abort_identifier, reason_code	This is an FAL internal primitive used to convey an Abort request primitive from the FSPM to the ARPM.
UCS_req	FSPM	vcr_id, service_type, user_data	809) This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) request primitive from the FSPM to the ARPM.

Table 137 – Primitives issued by ARPM to FSPM

Primitive name	Source	Associated parameters	Functions
EST_cnf(+)	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish response(+) primitive from the ARPM to the FSPM.
Abort_ind	ARPM	vcr_id, abort_detail, abort_identifier, reason_code	This is an FAL internal primitive used to convey an Abort primitive from the ARPM to the FSPM.
UCS_ind	ARPM	vcr_id, service_type, user_data	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) indication primitive from the ARPM to the FSPM.

4.8.2.2.2 Parameters of FSPM/ARPM primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in Table 138.

Table 138 – Parameters used with primitives exchanged between FSPM and ARPM

Parameter name	Description
arep_id	This parameter is used to unambiguously identify an instance of the AREP that has issued a primitive. A means for such identification is not specified by this standard.
vcr_id	This parameter is used to unambiguously identify an instance of the VCR that issued a primitive or that is the primitive destination. A means for such identification is not specified by this standard.
user_data	This parameter conveys AR-User data (Including the service specific parameters).
abort_detail	This parameter conveys the value that is used for the Abort_Detail parameter.
abort Identifier	This parameter conveys the value that is used for the Identifier parameter.
reason_code	This parameter conveys the value that is used for the Reason_Code parameter.
remote_address	This parameter conveys the remote address of the underlying layer.
service_type	This parameter conveys the protocol Id and the service within the protocol.

4.8.2.3 DLL mapping of MulticastAREP class

4.8.2.3.1 Formal definition

Subclause 4.8.2.3 describes the mapping of the Publisher / Subscriber and Report Distribution AREP Classes to the Socket model. The Socket mapping attributes and their permitted values used with the Publisher / Subscriber and Report Distribution AREP classes are defined in 4.8.2.3.

CLASS: **Multicast**
PARENT CLASS: AR Endpoint
ATTRIBUTES:

1	(m)	KeyAttribute:	LocalAndRemoteAddresses
1.1	(m)	KeyAttribute:	LocalAddress
1.2	(m)	KeyAttribute:	RemoteAddress
2	(m)	Attribute:	TransferType (CONN, CNLS)
4	(c)	Constraint:	Role = SUBSCRIBER | REPORT SINK
4.1	(m)	Attribute:	GuardBand
5	(c)	Constraint:	Role = PUBLISHER | REPORT SOURCE
5.1	(m)	Attribute:	TransmitDelayTime
6	(m)	Attribute:	BufferSize
7	(m)	Attribute:	HdrOptions
8	(c)	Constraint:	Role = SUBSCRIBER | REPORT SINK
8.1	(m)	Attribute:	MaxApduLength

4.8.2.3.2 Attributes

LocalAndRemoteAddresses

This key attribute identifies the AREP. It is composed of the local and remote addresses of the underlying layer for the AREP.

TransferType

This attribute specifies whether the connection-oriented (CONN) or connectionless (CNLS) services of the underlying layer are used by the AREP.

GuardBand

This conditional attribute specifies the lower and upper bound for the field being protected from rollover. It is present if the ROLE is SUBSCRIBER or REPORT SINK.

TransmitDelayTime

This conditional attribute specifies the amount of time that a sending session endpoint accumulates messages to send in its buffer. It is present if the ROLE is PUBLISHER or REPORT SOURCE. The time period starts when the first message is placed into the transmit buffer after either the buffer has been initialized or after it has been emptied by the previous transmission.

If the endpoint has no messages to send (the buffer is empty) and it receives a message to send, it loads the message into the buffer and waits for additional messages to concatenate into the buffer. If the buffer fills before the Transmit Delay Time has been reached, the Type 5 FAL AE stops the timer and sends the buffer. When this time interval has expired and the buffer has not yet filled, it sends the contents of the buffer.

BufferSize

This attribute specifies the size in octets of the session endpoint's send buffer. The buffer holds an integral number of messages. If a message is received for transfer that would overflow the buffer, the buffer is sent first, and the message is then added as the first message in the buffer.

HdrOptions

This attribute specifies which Message Trailer fields are present in the messages sent and received on the session.

MaxApduLength

This conditional attribute specifies the maximum permitted length in octets of APDUs sent and received on the AR. It is present if the ROLE is SUBSCRIBER or REPORT SINK.

4.8.2.4 MulticastARPM state machine

4.8.2.4.1 MulticastARPM states

The defined states and their descriptions of the Publisher / Subscriber ARPM are shown in Table 139 and Figure 3. For CONN, the closed state is never left until the connection has been established. How this connection is established is implementation dependent.

Table 139 – Publisher / subscriber ARPM states

CLOSED	The AREP is defined, but not capable of sending or receiving arbitrary FAL-PDUs. It may send or receive Establish service FAL-PDUs while in this state.
OPEN	The AREP is defined and capable of sending or receiving FAL-PDUs.

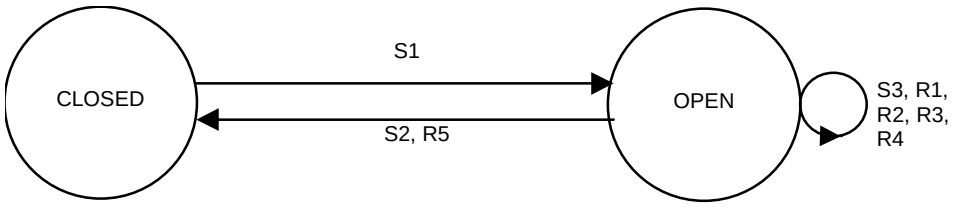


Figure 3 – State transition diagram of the publisher / subscriber ARPM

4.8.2.4.2 Multicast ARPM state table

Table 140 and Table 141 define the Multicast ARPM state machine.

Table 140 – MulticastARPM state table – sender transitions

#	Current state	**Event** or **condition** **=> action**	Next state
810) S1	811) CLOSED	812) EST_req 813) => 814) EST_cnf(+){ 815) arep_id := GetArepId (), 816) user_data := “null” 817) }	818) OPEN
819) S2	820) OPEN 821)	822) Abort_req 823) =>	824) CLOSED
825) S3	826) OPEN	827) UCS_req 828) && Role = “PUBLISHER” || "REPORT SOURCE" 829) => 830) FAL-PDU_req { 831) dmpm_service_name := "SMPM_Send", 832) arep_id := GetArepId (), remote_address := RemoteAddress*, 833) fal_pdu := BuildFAL-ReqRspPDU (fal_service_type = service_type, 834) fal_data := user_data) 835) }	836) OPEN

Table 141 – MulticastARPM state table – receiver transitions

#	Current state	**Event** or **condition** **=> action**	Next state
837) R1	838) OPEN	839) FAL-PDU_ind 840) && FAL_Pdu_Valid(fal_pdu) = False 841) => 842) Report to local management	843) OPEN
844) R2	845) OPEN	846) FAL-PDU_ind 847) && Endpoint Type = “SUBSCRIBER” || "REPORT SINK" 848) && FAL_Pdu_Confirmed(fal_pdu) = True 849) => 850) Report to local management	851) OPEN
852) R3	853) OPEN	854) FAL-PDU_ind 855) && Endpoint Type = “SUBSCRIBER” 856) && FAL_Pdu_GetVcrId() <> NULL 857) => 858) UCS_ind* { 859) arep_id = GetArepId (), 860) vcr_id = GetVcrId (), 861) service_type = GetServiceType(fal_pdu), 862) user_data = GetUserData(fal_pdu) 863) } 864) 865) * Delivered to each VCR that subscribes to the FDA Address in the APDU Header	866) OPEN
867) R4	868) OPEN	869) FAL-PDU_ind 870) && Endpoint Type = “REPORT SINK” 871) => 872) UCS_ind* { 873) arep_id = GetArepId (), 874) vcr_id = GetVcrId (), 875) service_type = GetServiceType(fal_pdu), 876) user_data = GetUserData(fal_pdu) 877) } 878) 879) * Delivered to all VCRs associated with this AREP	880) OPEN
881) R5	882) OPEN	883) ErrorToARPM 884) && ErrorType = “socket disconnect” 885) => 886) Abort_ind* { 887) arep_id = GetArepId (), 888) locally_generated = "True", 889) identifier = "FAL", 890) reason_code = "Socket disconnected" 891) } 892) 893) * Delivered to all VCR associated with this AREP	894) CLOSED

4.8.3 Functions used by ARPMs

Table 142 through Table 158 define the functions used by the ARPMs.

4.8.3.1 BuildFAL-ErrPDU()

Table 142 – BuildFAL-ErrPDU()

Function	
Returns a fal_pdu that is an error response with the specified errors. The request_pdu input parameter provides the necessary information to construct the appropriate error response pdu.	
Input parameters	Output parameters
request_pdu error_class error_code	fal_pdu

4.8.3.2 BuildFAL-ReqRspPDU()

Table 143 – BuildFAL-ReqRspPDU()

Function	
Builds an FAL PDU out of the parameters given as input variables. This includes: Building the FAL pdu header Adding the fal_data (AR user data) Generating the FAL-PDU trailer	
Input parameters	Output parameters
service_type fal_data	fal_pdu

4.8.3.3 GetArepId()

Table 144 – GetArepId()

Function	
Returns a value that can unambiguously identify the current AREP.	
Input parameters	Output parameters
None	AREP Identifier

4.8.3.4 ConfigurationArCheckOK()

Table 145 – ConfigurationArCheckOK()

Function	
Returns True if: The request fal_pdu is not a configuration request The request fal_pdu is a configuration request and there are no other configuration ARs established	
Input parameters	Output parameters
fal_pdu	True | False

4.8.3.5 FAL_Pdu_BufferSize()

Table 146 – FAL_Pdu_BufferSize()

Function	
This function returns the buffer size from the AR Establish fal_pdu.	
Input parameters	Output parameters
fal_pdu	BufferSize

4.8.3.6 FAL_Pdu_Confirmed()

Table 147 – FAL_Pdu_Confirmed()

Function	
This function returns true if the fal_pdu is confirmed	
Input parameters	Output parameters
fal_pdu	True | False

4.8.3.7 FAL_Pdu_DuplicateMsg ()

Table 148 – FAL_Pdu_DuplicateMsg ()

Function	
This function returns True if the received fal_pdu is determined to be a duplicate based on its message number and the vcr_id parameter. Otherwise, it returns False.	
Input parameters	Output parameters
vcr_id, fal_pdu	True | False

4.8.3.8 FAL_Pdu_GetVcrId()

Table 149 – FAL_Pdu_GetVcrId()

Function	
Using the FDA Address in the fal_pdu header, returns a value that identifies the destination VCR endpoint associated with the AR. May return a list if the AREP is a subscriber. Returns NULL if the FDA Address does not identify a VCR endpoint.	
Input parameters	Output parameters
fal_pdu	VCR Identifier

4.8.3.9 FAL_Pdu_InactivityCloseTime()

Table 150 – FAL_Pdu_InactivityCloseTime()

Function	
This function returns the InactivityCloseTime from the AR Establish fal_pdu.	
Input parameters	Output parameters
AR Establish fal_pdu	InactivityCloseTime

4.8.3.10 FAL_Pdu_TransmitDelayTime()

Table 151 – FAL_Pdu_TransmitDelayTime()

Function	
This function returns the TransmitDelayTime from the AR Establish fal_pdu.	
Input parameters	Output parameters
AR Establish fal_pdu	TransmitDelayTime

4.8.3.11 FAL_Pdu_SvcType()

Table 152 – FAL_Pdu_SvcType()

Function	
This function decodes the FAL-PDU that is conveyed in the fal_pdu parameter and retrieves the FAL-PDU service type.	
Input parameters	Output parameters
fal_pdu	service type

4.8.3.12 FAL_Pdu_RemoteAddress()

Table 153 – FAL_Pdu_RemoteAddress()

Function	
Returns the address of the sender of the received fal_pdu.	
Input parameters	Output parameters
fal_pdu	address of sender

4.8.3.13 FAL_Pdu_TrailerFields()

Table 154 – FAL_Pdu_TrailerFields()

Function	
This function returns the trailer fields of the fal_pdu.	
Input parameters	Output parameters
fal_pdu	Trailer fields present in the FAL-PDU.

4.8.3.14 FAL_Pdu_ServiceSpecificParameters()

Table 155 – FAL_Pdu_ServiceSpecificParameters()

Function	
This function returns the service specific parameters, including user data, from an fal_pdu.	
Input parameters	Output parameters
fal_pdu	Service Specific parameters

4.8.3.15 FAL_Pdu_Valid()

Table 156 – FAL_Pdu_Valid()

Function	
The function returns False if the ARPM can determine that: 1. the fal_pdu header or trailer contains invalid values or inconsistent values, or 2. the fal_pdu header or trailer contains options that do not match those negotiated for the AR, 3. the fal_pdu cannot be properly decoded or identified. The Service Specific parameters are not examined by this function.	
Input parameters	Output parameters
fal_pdu	True | False

4.8.3.16 MaxOutstandingReached()

Table 157 – MaxOutstandingReached()

Function	
The function tests the local counter for the current number of outstanding requests to determine if it has reached the limit defined by the MaxOutstanding session attribute. It returns True if the limit has been reached. If not, it increments the local counter for the current number of outstanding requests and returns False.	
Input parameters	Output parameters
None	True | False

4.8.3.17 StartInactivityCloseTimer()

Table 158 – StartInactivityCloseTimer()

Function	
The function (re)starts a timer for the period of time defined by the Inactivity Close Time.	
Input parameters	Output parameters
None	None

4.9 DLL mapping protocol machine (DMPM)

4.9.1 General

The DLL mapping is represented by the Socket model.

The Socket model defines an abstract service interface that permits the underlying layer to exist at the data link layer, network layer, or transport layer. Two types of services are supported by the Socket model, connection oriented (CONN) and connectionless (CNLS).

4.9.2 Primitive definitions

4.9.2.1 Primitives exchanged between DMPM and ARPM

Table 159 and Table 160 define the primitives exchanged between the DMPM and the ARPM.

Table 159 – Primitives issued by ARPM to DMPM

Primitive name	Source	Associated parameters	Functions
FAL-PDU_req	ARPM	smpm_service_name, remote_address, fal_pdu	This primitive is used request the DMPM to transfer an FAL-PDU. The list of smpm_service_names is: SMPM_Send

Table 160 – Primitives issued by DMPM to ARPM

Primitive name	Source	Associated parameters	Functions
FAL-PDU_ind	DMPM	smpm_service_name, remote_address, fal_pdu	This primitive is used to pass an FAL-PDU received as a Socket model service data unit to a designated ARPM. It also carries some of the Socket model parameters that are referenced in the ARPM. The list of smpm_service_names is: SMPM_Receive
ErrorToARPM	DMPM	originator, reason	This primitive is used to convey selected communication errors reported by the Socket model to a designated ARPM.

4.9.2.2 Parameters of ARPM/DMPM primitives

The parameters used with the primitives exchanged between the ARPM and the DMPM are described in Table 161.

Table 161 – Parameters used with primitives exchanged between ARPM and DMPM

Parameter name	Description
fal_pdu	This parameter conveys the value of the socket_user_data parameter.
smpm_service_name	This parameter conveys a Socket model service name.
remote_address	This parameter conveys the value of the remote address of the underlying layer.
originator	This parameter identifies the entity that detected the error that is reported using the ErrorToARPM primitive.
reason	This parameter identifies the cause of the error that is reported using the ErrorToARPM primitive.

4.9.2.3 Primitives exchanged between the socket model and DMPM

See Table 162 for the definitions.

NOTE 1 The following primitives and their parameters are exchanged between the Socket model and the DMPM.

NOTE 2 The DMPM instance uses either CNLS or CONN services.

NOTE 3 It is assumed that each ARPM is bound to the local address and therefore this is never an argument with any primitive.

Table 162 – Primitives exchanged between the socket model and DMPM

Primitive name	Source	Associated parameters
socket_send.ind	Socket model	socket_user_data
socket_sendto.ind	Socket model	socket_remote_address, socket_user_data
socket_sendto.req	DMPM	socket_remote_address, socket_user_data
socket_send.req	DMPM	socket_user_data

4.9.2.4 Parameters of DMPM/socket model primitives

See Table 163 for the definitions.

Table 163 – Parameters of DMPM/socket model primitives

Parameter name	Description
socket_remote_address	This parameter conveys the value of the remote Socket model address.
socket_user_data	The complete message that the socket model shall send to / received from specified destination

4.9.3 DMPM state machine

4.9.3.1 DMPM states

The defined state of the DMPM together with its description is listed in Table 164 and Figure 4.

Table 164 – DMPM state descriptions

State Name	Description
ACTIVE	The DMPM in the ACTIVE state is ready to transmit or receive primitives to or from the Socket model and the ARPM.

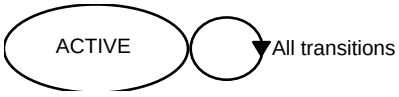


Figure 4 – State transition diagram of DMPM

4.9.3.2 DMPM state table

Table 165 and Table 166 define the DMPM state machine.

NOTE 1 Although each primitive contains all the available parameters, only those applicable to particular ARPM are relevant.

NOTE 2 Parameters starting with a capital letter, “TransmitDelayTime ” for instance, refer to those defined in the attribute list of each ARPM. Therefore, they are not conveyed by the service primitives defined here.

Table 165 – DMPM state table – sender transitions

#	Current state	Event or condition => action	Next state
895) S1	896) ACTIVE	897) FDA-PDU_req 898) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "FIRST MESSAGE" 899) => 900) LoadBuffer(arep_id, fal_pdu) 901) StartTransmitDelayTimer(arep_id)	902) ACTIVE
903) S2	904) ACTIVE	905) FDA-PDU_req 906) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "LESS THAN" 907) => 908) LoadBuffer(arep_id, fal_pdu)	909) ACTIVE
910) S3	911) ACTIVE	912) FDA-PDU_req 913) && ConnectionOriented() = True 914) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "EQUAL" 915) => 916) socket_send.req { 917) user_data = GetBufferedData(arep_id) 918) }	919) ACTIVE
920) S4	921) ACTIVE	922) FDA-PDU_req 923) && ConnectionOriented() = FALSE 924) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "EQUAL" 925) => 926) socket_sendto.req { 927) remote_address = remote_address, 928) user_data = GetBufferedData(arep_id) 929) }	930) ACTIVE
931) S5	932) ACTIVE	933) FDA-PDU_req 934) && ConnectionOriented() = True 935) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "GREATER THAN" 936) => 937) socket_send.req { 938) user_data = fal_pdu 939) } 940) LoadBuffer(arep_id,fal_pdu) 941) StartTransmitDelayTimer(arep_id)	942) ACTIVE
943) S6	944) ACTIVE	945) FDA-PDU_req 946) &&ConnectionOriented()=FALSE 947) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "GREATER THAN" 948) => 949) socket_sendto.req { 950) remote_address = remote_address, 951) user_data = GetBufferedData(arep_id) 952) } 953) LoadBuffer(arep_id, fal_pdu) 954) StartTransmitDelayTimer(arep_id)	955) ACTIVE
956) S7	957) ACTIVE	958) Transmit delay timer expires 959) && ConnectionOriented = False 960) => 961) socket_sendto.req { 962) remote_address = GetRemoteAddress(arep_id), 963) user_data = GetBufferedData(arep_id) 964) }	965) ACTIVE
966) S8	967) ACTIVE	968) Transmit delay timer expires 969) && ConnectionOriented() = True 970) => 971) socket_send.req { 972) user_data = GetBufferedData(arep_id) 973) }	974) ACTIVE

Table 166 – DMPM state table – receiver transitions

#	Current state	Event or condition => action	Next state
975) R1	976) ACTIVE	977) socket_sendto.ind 978) => 979) FAL-PDU_ind { 980) smpm_service_name := "SMPM_Receive", 981) remote_address := socket_remote_address, 982) fal_pdu := socket_user_data 983) }	984) ACTIVE
985) R2	986) ACTIVE	987) socket_send.ind 988) => 989) FAL-PDU_ind { 990) smpm_service_name := "SMPM_Receive", remote_address := GetConnectionId(arep_id) , 991) fal_pdu := socket_user_data 992) }	993) ACTIVE
994) R3	995) ACTIVE	996) “Connection breaks” 997) => 998) ErrorToArpm { 999) originator= local_socket, 1000) reason = “Socket disconnected” 1001) }	1002) ACTIVE

4.9.3.3 Functions used by DMPM

Table 167 through Table 172 define the functions used by the DMPM.

4.9.3.3.1 Function ConnectionOriented

Table 167 – ConnectionOriented

Name	ConnectionOriented	Used in	ARPM
Input		Output	
		True | False	
Function			
1003) This function returns True if the SMPM instance is connection oriented (CONN) and False if the SMPM instance in connection less (CNLS).			

4.9.3.3.2 Function GetBufferedData

Table 168 – GetBufferedData

Function	
This function returns the contents of the AREP send buffer, and marks the buffer as empty.	
Input parameters	Output parameters
ArepId	buffered_data

4.9.3.3.3 Function GetConnectionId

Table 169 – GetConnectionId

Function	
This function returns the connection id of a connection-oriented socket.	
Input parameters	Output parameters
ArepId	remote_address

4.9.3.3.4 Function LoadBuffer

Table 170 – LoadBuffer

Function	
This function concatenates the user_data into the AREP buffer	
Input parameters	Output parameters
ArepId, user_data	None

4.9.3.3.5 Function RemainingBufferSizeCheck

Table 171 – RemainingBufferSizeCheck

Function	
This function compares the size of the user_data with the size of the space remaining in the AREP send buffer. It returns: "FIRST MESSAGE" the buffer is currently empty and the size of the user_data less than the size of the buffer. "EQUAL" the size of the user_data is the same size as the space remaining in the buffer. "LESS THAN" the size of the user_data is less than the size of the space remaining in the buffer. "GREATER THAN" the size of the user_data is greater than the size of the space remaining in the buffer.	
Input parameters	Output parameters
ArepId, user_data	buffer_status

4.9.3.3.6 StartTransmitDelayTimer

Table 172 – StartTransmitDelayTimer

Function	
This function starts or restarts the Transmit Delay Timer of the AREP identified by the ArepId parameter	
Input parameters	Output parameters
ArepId	None

Bibliography

IEC 61158-3-1, *Industrial communication networks – Fieldbus specifications – Part 3-1: Data-link layer service definition – Type 1 elements*

IEC 61158-4-1, *Industrial communication networks – Fieldbus specifications – Part 4-1: Data-link layer protocol specification – Type 1 elements*

IEC 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 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*[1]

_____

[1] Withdrawn.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-5: Spécification du protocole 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 attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

L'attention est attirée sur le fait que l'utilisation du type de protocole associé est restreinte par les détenteurs des droits de propriété intellectuelle. En tout état de cause, l'engagement de renonciation partielle aux droits de propriété intellectuelle pris par les détenteurs de ces droits autorise l'utilisation d'un type de protocole de couche avec les autres protocoles de couche du même type, ou dans des combinaisons avec d'autres types autorisées explicitement par les détenteurs des droits de propriété intellectuelle pour ce type..

NOTE Les combinaisons de types de protocoles sont spécifiées dans la CEI 61784-1 et la CEI 61784-2.

La Norme internationale CEI 61158-6-5 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 deuxième édition annule et remplace la première édition parue en 2007. Cette édition constitue une révision technique.

Les modifications majeures par rapport à l'édition précédente sont énumérées ci-dessous:

- Support ajouté pour le remplissage de messages
- Règles de codage clarifiées
- Service clarifié de session ouverte
- Synchronisation de temps désormais présente dans un message annonciateur
- Options de redondance supplémentaires dans un message annonciateur

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65C/764/FDIS	65C/774/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.

INTRODUCTION

La présente partie de la CEI 61158 est l'une de la série élaborée afin de faciliter l'interconnexion des composants de systèmes d'automation. Elle est liée à d'autres normes dans l'ensemble tel que défini par le modèle de référence des bus de terrain "à trois couches" décrit dans la CEI 61158-1.

Le protocole d'application fournit le service d'application en utilisant les services disponibles issus de la couche liaison de données ou d'une autre couche immédiatement inférieure. Le but principal de la présente norme est de fournir un ensemble de règles pour la communication exprimées en termes des procédures devant être accomplies par des entités d'application (AE) d'homologues au moment de la communication. Ces règles pour la communication visent à fournir une base solide pour le développement et de servir une diversité de besoins:

- comme un guide pour les réalisateurs et les concepteurs;
- pour une utilisation dans les essais et achats d'équipements;
- comme partie intégrante d'un accord pour l'admission de systèmes dans l'environnement de systèmes ouverts;
- comme affinement pour la compréhension de communications prioritaires au sein de l'OSI (Open Systems Interconnexion, c'est-à-dire Interconnexion des systèmes ouverts).

La présente norme est concernée, en particulier, par la communication et l'interfonctionnement des capteurs, des effecteurs et autres appareils d'automatisation. L'utilisation de la présente norme conjointement à d'autres normes positionnées dans les modèles de référence de l'OSI ou de bus de terrain permet à n'importe quelle combinaison de systèmes autrement incompatibles de fonctionner.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-5: Spécification du protocole 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 des éléments communs pour les communications par messagerie de base prioritaire et non prioritaire entre des programmes d'application dans un environnement d'automation et des matériaux spécifiques aux bus de terrain de Type 5. Le terme «prioritaire» est utilisé pour montrer la présence d'une fenêtre temporelle, dans les limites de laquelle une ou plusieurs actions spécifiées sont tenues d'être parachevées avec un niveau défini de certitude. Le manquement à parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent les actions, avec le risque concomitant pour l'équipement, l'installation et éventuellement pour la vie humaine.

La présente norme définit de manière abstraite le comportement visible de l'extérieur fourni par la couche application de bus de terrain de Type 5 en termes

a) de la syntaxe abstraite définissant les unités de données du protocole de la couche application acheminées entre les entités d'application engagées dans une communication;

b) de la syntaxe de transfert définissant les unités de données du protocole de la couche application entre les entités d'application engagées dans une communication;

c) du diagramme d'états abstrait dans un contexte d'application définissant le comportement d'un service d'application visible entre les entités d'application engagées dans une communication; et

d) des diagrammes d'états de relations entre applications définissant le comportement de communication visible entre les entités d'application engagées dans une communication.

La présente norme a pour objectif de définir le protocole fourni pour

1) définir la représentation sur le câble des primitives de service définies dans la CEI 61158-5-5, et

2) définir le comportement visible de l'extérieur et associé à leur transfert.

La présente norme spécifie la couche application de bus de terrain de Type 5 de la CEI en conformité avec le Modèle de référence de base pour l'interconnexion des systèmes ouverts (ISO/CEI 7498-1) et la structure de la couche application OSI (ISO/CEI 9545).

Des services et protocoles de FAL sont fournis par les entités d'application (AE) de FAL contenues au sein des processus d'application. Une AE de FAL est composée d'un ensemble d'éléments de services d'application (ASE) orientés objet et d'une entité de gestion de couche (LME) qui gère l'AE. Les ASE fournissent des services de communication qui fonctionnement sur un ensemble des classes connexes d'objets des processus d'application (APO). L'un des ASE de FAL est un ASE de gestion qui fournit un ensemble typique de services pour la gestion des instances des classes de FAL.

Bien que ces services spécifient, du point de vue des applications, comment les demandes et les réponses sont émises et livrées, ils ne comprennent pas la spécification de ce que les applications de demande et de réponse sont tenues d'en faire. Cela veut dire que les aspects comportementaux des applications ne sont pas spécifiés; seule la définition des demandes et des réponses, qu'ils peuvent envoyer/recevoir, est spécifiée. Ainsi, les utilisateurs de FAL sont dotés d'une plus grande flexibilité pour la normalisation d'un tel comportement d'objet. Outre ces services, des services de support, également définis dans la présente norme, donnent accès à la FAL pour le contrôle de certains aspects de son fonctionnement.

1.2 Spécifications

Le but principal de la présente norme est de spécifier la syntaxe et le comportement du protocole de couche application qui achemine les services de couche application définis dans la CEI 61158-5-5.

Un autre objet consiste à assurer des trajets de migration à partir des protocoles de communications industrielles préexistants. C'est ce dernier objectif qui donne lieu à la diversité des protocoles normalisés dans la série CEI 61158-6.

1.3 Conformité

La présente norme ni ne spécifie de mises en œuvre individuelles ou de produits individuels ni ne contraint les mises en œuvre d'entités de couche application au sein des systèmes d'automatisation industriels. La conformité est obtenue par la mise en œuvre de cette spécification de protocole de couche application.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

NOTE Toutes les parties de la série CEI 61158, ainsi que la CEI 61784-1 et la CEI 61784-2 font l'objet d'une maintenance simultanée. Les références croisées à ces documents dans le texte se rapportent par conséquent aux éditions datées dans la présente liste de références normatives.

CEI 61158-1, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Présentation et lignes directrices des séries CEI 61158 et CEI 61784*

CEI 61158-5-5, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-5 : Définition des services de la couche application – Éléments de type 5*

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/IEC 8825:1990, *Technologies de l'information -- Interconnexion de systèmes ouverts – Spécification de règles de base pour coder la notation de syntaxe abstraite numéro UNE (ASN.1)*[1]

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

[1] Retirée

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

IETF RFC 791, *Protocole Internet;* disponible à l'adresse <http://www.ietf.org>

3 Termes, définitions, symboles, abréviations et conventions

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

3.1 Termes et définitions d'autres normes ISO/CEI

3.1.1 Termes et définitions de l'ISO/CEI 7498-1

a) syntaxe abstraite
b) entité d'application
c) processus d'application
d) unité de données de protocole d'application
e) élément de service d'application
f) invocation d'entités d'application
g) invocation de processus d'application
h) transaction d'application
i) contexte de présentation
j) système ouvert réel
k) syntaxe de transfert

3.1.2 Termes et définitions de l'ISO/CEI 9545

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

3.1.3 Termes et définitions de l'ISO/CEI 8824

a) identificateur d'objet
b) type
c) valeur
d) type simple
e) type structuré
f) type composant
g) indicateur
h) type booléen
i) vrai
j) faux
k) type entier
l) type chaîne de bits
m) type chaîne d'octets
n) type non valide
o) type de séquence
p) séquence de type
q) type de choix
r) type marqué
s) tout type
t) module
u) production

3.1.4 Termes et définitions de l'ISO/CEI 8825

a) codage (d'une valeur de données)
b) valeur de données
c) octets d'identificateur (la présente norme utilise le singulier)
d) octet(s) de longueur (tant le singulier que le pluriel sont utilisés dans la présente norme)
e) octets de contenu

3.2 Termes de la CEI 61158-1

Pour les besoins du présent document, les termes suivants de la CEI 61158-1 s'appliquent.

3.2.1
application
fonction ou structure de données pour laquelle des données sont consommées ou produites

3.2.2
interopérabilité de couche application
capacité des entités d'application d'accomplir des opérations coordonnées et coopératives en utilisant les services de la FAL

3.2.3
objet d'application
classe d'objet qui gère et assure l'échange de messages pendant le mode exécution à travers le réseau et à l'intérieur de l'appareil de réseau

Note 1 à l'article: Plusieurs types de classes d'objets d'application peuvent être définis.

3.2.4
processus d'application
partie d'une application distribuée sur un réseau, située sur un appareil et associée à une adresse non ambiguë

3.2.5
identificateur de processus d'application
composant qui identifie de multiples processus d'application utilisés dans un appareil

3.2.6
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'objet 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 FAL Object Management (Gestion d'objet de la FAL). Les services de Gestion d'objet de la FAL peuvent être utilisés pour charger ou mettre à jour des définitions d'objet, pour lire des définitions d'objet, et pour créer et supprimer de manière dynamique des objets d'application et leurs définitions correspondantes.

3.2.7
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

3.2.8
relation entre applications
association coopérative entre deux ou plusieurs invocations d'entités d'application (application-entity-invocation) à des fins d'échange d'informations et de coordination de leur fonctionnement conjoint

Note 1 à l'article : Cette relation est activée soit par l'échange d'unités de données de protocole d'application, soit à la suite d'activités de préconfiguration.

3.2.9
élément de service d'application des relations 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

3.2.10
point d'extrémité de 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 de relation entre applications.

3.2.11
attribut
description d'une caractéristique ou fonction 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 de 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.

3.2.12
comportement
indication de la façon dont un objet réagit à des événements particuliers

Note 1 à l'article: Sa description comprend la relation entre les services et les valeurs d'attributs.

3.2.13
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.

3.2.14
attributs de classe
attribut qui est partagé par tous les objets au sein de la même classe

3.2.15
code de classe
identificateur unique attribué à chaque classe d'objets

3.2.16
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.

3.2.17
client
(a) objet qui utilise les services d'un autre objet (serveur) pour accomplir une tâche

(b) initiateur d'un message auquel un serveur réagit, par exemple, le rôle d'un point d'extrémité d'AR dans lequel il émet des APDU de demande de service confirmé vers un seul point d'extrémité d'AR agissant en tant que serveur

3.2.18
trajet d'acheminement
flux unidirectionnel des APDU à travers une relation entre applications

3.2.19
cyclique
terme utilisé pour la description d'événements qui se répètent d'une manière régulière et répétitive

3.2.20
AR dédiée
AR utilisée directement par l'utilisateur de FAL

Note 1 à l'article: Sur des AR dédiées, seules la tête de FAL et les données d'utilisateur sont transférées.

3.2.21
appareil
équipement matériel physique relié à la liaison

Note 1 à l'article: Un appareil peut contenir plus d'un nœud.

3.2.22
profil d'appareil
ensemble des informations et des fonctionnalités dépendantes de l'appareil assurant la cohérence entre les appareils similaires du même type d'appareil

3.2.23
AR dynamique
AR qui nécessite d'être mise dans un état établi par le biais des procédures d'établissement d'AR

3.2.24
point d'extrémité
une des entités en communication impliquées dans une connexion

3.2.25
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)

3.2.26
classe d'erreurs
regroupement général pour des définitions d'erreurs

Note 1 à l'article: Les codes d'erreur pour des erreurs spécifiques sont définis dans une classe d'erreurs.

3.2.27
code d'erreur
identification d'un type spécifique d'erreur dans une classe d'erreurs

3.2.28
sous-réseau de FAL
sous-réseaux constitués d'un ou plusieurs segments de liaison de données

Note 1 à l'article: Les sous-réseaux FAL sont autorisés à contenir des ponts, mais pas des routeurs. Ils sont identifiés par un sous-ensemble de l'adresse réseau.

3.2.29
appareil logique
certaine classe de FAL qui extrait un composant logiciel ou un composant de microprogramme (firmware) sous forme d'un appareil intégré et autonome au sein d'un appareil d'automation

3.2.30
informations de gestion
information accessible par le réseau qui soutient la gestion de l'opération du système de bus de terrain, y compris la couche application

Note 1 à l'article: La gestion comprend des fonctions telles que commande, surveillance et diagnostic.

3.2.31
réseau
ensemble de nœuds reliés par un certain type de support de communication

Note 1 à l'article: Les trajets de connexion entre toute paire de nœuds peuvent comporter des répétiteurs, des routeurs et des passerelles.

3.2.32
homologue
rôle d'un point d'extrémité d'une AR dans lequel il est capable d'agir en tant que client et serveur

3.2.33
point d'extrémité d'AR prédéfini
point d'extrémité d'AR qui est défini localement au sein d'un appareil sans utiliser le service de création

Note 1 à l'article: Des AR prédéfinies et non préétablies sont installées avant d'être utilisées.

3.2.34
point d'extrémité d'AR préétabli
point d'extrémité d'AR dans un état établi au cours de la configuration des AE qui contrôlent ses points d'extrémité

3.2.35
éditeur
rôle d'un point d'extrémité d'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 spécialisée. Deux types d'éditeurs sont définis par la présente norme, "Pull Publisher" et "Push Publisher", chacun de ceux-ci étant défini séparément.

3.2.36
serveur
a) rôle d'un AREP dans lequel il retourne une APDU de réponse de service confirmée au client qui a émis la demande

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

3.2.37
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

Note 1 à l'article: Un ensemble de services communs est défini et des dispositions permettant la définition de services propres à un objet sont fournies. Des services spécifiques à un objet sont des services définis par une classe d'objets particulière pour effectuer une fonction demandée qui n'est pas effectuée par un service typique.

3.2.38
abonné

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

Note 1 à l'article: Deux types d'abonnés sont définis par la présente norme: les "pull subscribers" 'abonnés par extraction) et les "push subscribers" (abonnés par émission); chacun faisant l'objet d'une définition distincte.

3.3 Abréviations et symboles

AE	Application Entity (Entité d'application)
AL	Application Layer (Couche application)
ALME	Application Layer Management Entity (Entité de gestion de couche application)
ALP	Application Layer Protocol (Protocole de couche application)
APO	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 (Identificateur de processus d'application)
AR	Application Relationship (Relation entre applications)
AREP	Application Relationship End Point (Point d'extrémité de relation entre applications)
ASCII	American Standard Code for Information Interchange (Code de normalisation américain pour l'échange d'informations)
ASE	Application Service Element (Élément de service d'application)
Cnf	Confirmation
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 de connexion de liaison de données)
DLL	Data-link layer (Couche liaison de données)
DLM	Data-link-management (Gestion de liaison de données)
DLSAP	Data-link Service Access Point (Point d'accès au service de liaison de données)
DLSDU	DL-service-data-unit (Unité de données de service de DL)
FAL	Fieldbus Application Layer (Couche application de bus de terrain)
ID	Identificateur
CEI	Commission Électrotechnique Internationale
Ind	Indication
LME	Layer Management Entity (Entité de gestion de couche)
OSI	Open Systems Interconnect (Interconnexion de systèmes ouverts)
QoS	Quality of Service (Qualité de service)
Req	Request (Demande)
Rsp	Response (Réponse)

SAP Service Access Point (Point d'accès au service)

SDU Service Data Unit (Unité de données de service)

SMIB System Management Information Base (Base d'informations de gestion de systèmes)

SMK System Management Kernel (Noyau de gestion de systèmes)

VFD Virtual Field Device (Appareil de champ virtuel)

3.4 Conventions

3.4.1 Concept général

La FAL est définie comme un ensemble des ASE orientés objet. Chaque ASE est spécifié dans un paragraphe séparé. Chaque spécification d'ASE est divisée en trois parties: ses définitions de la classe, ses services et sa spécification des protocoles. Les deux premières parties sont contenues dans la série CEI 61158-5. La spécification des protocoles pour chacun des ASE est définie dans la présente norme.

Les définitions de classe définissent les attributs des classes prises en charge par chaque ASE. Les attributs sont accessibles à partir des instances de la classe en utilisant les services d'ASE de gestion spécifiés dans la CEI 61158-5. La spécification des services définit les services qui sont fournis par l'ASE.

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

3.4.2 Conventions relatives aux définitions de classe

Les définitions du mapping de la couche liaison de données sont décrites à l'aide des modèles. Chaque modèle est composé d'une liste d'attributs pour une classe. La forme générale du modèle est définie dans la CEI 61158-5-5.

3.4.3 Conventions de syntaxe abstraite

Lorsque l'on utilise le paramètre "optionalParameterMap", un nombre de bits qui correspond à chaque production OPTIONAL (FACULTATIVE) ou DEFAULT (PAR DEFAUT) est donné sous forme d'un commentaire.

3.5 Conventions utilisées dans les diagrammes d'états

Les diagrammes d'états sont décrits dans le Tableau 1.

Tableau 1 – Conventions utilisées pour les diagrammes d'états

#	État courant	Événement /condition => action	État suivant
Nom de la transition.	L'État courant auquel s'applique cette transition d'état	Événements ou conditions qui déclenchent cette transition d'état. => Les actions effectuées lorsque les événements ou les conditions ci-dessus correspondent. Les actions figurent toujours au-dessous des événements ou des conditions	L'état suivant dans lequel passe le diagramme d'états une fois les actions effectuées.

Les conventions utilisées dans les diagrammes d'états sont représentées comme suit:

:= La valeur d'un objet à gauche est remplacée par la valeur d'un objet à droite. Si un objet à droite est un paramètre, il provient de la primitive représentée sous forme d'un événement d'entrée.

xxx Nom d'un paramètre.

Exemple :
Identifier := reason
1) signifie qu'une valeur d'un paramètre 'reason' est attribuée à un paramètre appelé 'Identifier'.

"xxx" Indique une valeur fixe.

Exemple :
Identifier := "abc"
2) signifie qu'une valeur "abc" est attribuée à un paramètre appelé 'Identifier'.
3)

= Une condition logique pour indiquer qu'un objet à gauche est égal à un objet à droite.

< Une condition logique pour indiquer qu'un objet à gauche est inférieur à l'objet à droite.

> Une condition logique pour indiquer qu'un objet à gauche est supérieur à l'objet à droite.

<> Une condition logique pour indiquer qu'un objet à gauche n'est pas égal à un objet à droite.

&& indique le "ET" logique.

|| indique le "OU" logique.

Ce concept permet d'exécuter une séquence d'actions en boucle au sein d'une transition. La boucle est effectuée pour toutes les valeurs comprises entre start_value et end_value.

Exemple :
for (Identifier := start_value to end_value)
actions
4) endfor

Ce concept permet d'exécuter d'autres actions en fonction de certaines conditions (la valeur d'un certain identificateur ou le résultat d'une action précédente) au sein d'une transition.

Exemple :
If (condition)
actions
else
actions
5) endif

Il est fortement recommandé de faire référence aux paragraphes concernant les définitions d'attributs d'AREP, les fonctions locales et les définitions de FAL-PDU pour comprendre le fonctionnement des diagrammes d'états. On suppose que les lecteurs possèdent une connaissance suffisante de ces définitions et savent les utiliser sans explications complémentaires.

4 Protocole

4.1 Vue d'ensemble

Les services et protocoles de Type 5 sont fournis par l'AE de Type 5 dénommée *Field Device Access (FDA) Agent* tout au long de cette spécification.

L'Agent de FDA achemine les APDU de Type 9 à l'aide des rapports d'application de Type 9 dénommés *FDA Sessions*, ou plus simplement *sessions,* tout au long de cette spécification de Type 5.

L'Agent de FDA peut fournir des fonctions de passerelle qui assurent le mapping entre les services de Type 5 et les services correspondants fonctionnant sur une DLL de Type "X". Des réseaux qui utilisent cette DLL de Type "X" sont dénommés *réseaux de Type "X"*.

4.2 Description de la syntaxe de FAL

4.2.1 Syntaxe abstraite de PDU

Combinée avec leur syntaxe de transfert, la syntaxe abstraite des APDU est spécifiée en 4.3.

4.2.2 Syntaxe abstraite des types de données

Combinée avec leur syntaxe de transfert, la syntaxe abstraite des types de données est spécifiée en 4.3.

4.3 Syntaxe de transfert

4.3.1 Généralités

La syntaxe de transfert combine la spécification de la syntaxe abstraite avec leurs codages sous forme d'un ensemble des APDU de format fixe. Chaque APDU contient un en-tête et un corps. Une fin de trame facultative est également définie. La présence d'une fin de trame est indiquée dans un en-tête d'APDU.

Le format de l'en-tête et de la fin de trame d'APDU est spécifié en 4.3.3 et 4.3.4. L'en-tête d'APDU contient des champs typiques pour tous les messages obligatoires. L'en-tête d'APDU contient des champs typiques pour toutes les APDU facultatives.

Les formats des corps d'APDU sont spécifiés en 4.3.5. Si une APDU de demande ou de réponse de services n'a pas de paramètres, cela est indiqué dans le paragraphe qui décrit le corps d'APDU. Le paragraphe correspondant n'est pas présent, si le service n'a pas de corps d'APDU de réponse ou d'erreur qui lui est associé. Les paramètres d'erreurs typiques sont utilisés dans de nombreuses APDU spécifiques à un service. Leurs définitions sont spécifiées en 4.3.6.1.

Les données d'utilisateur, lorsque présentes dans une APDU, sont définies comme un paramètre de corps d'APDU spécifique à un service. Le contenu et la longueur des données d'utilisateur dépendent du service et de l'objet accessibles. C'est pourquoi, ni l'un ni l'autre ne sont spécifiés comme partie intégrante du corps d'APDU. Cependant, la longueur des données d'utilisateur peut être calculée à partir de la longueur d'APDU contenue dans l'en-tête d'APDU.

L'en-tête d'APDU, la fin de trame et les corps sont composés d'un sous-ensemble de data types (c'est-à-dire types de données) définis dans la CEI 61158-5. Les codages pour ces data types sont définis ci-après et suivis des définitions de l'en-tête, de la fin de trame et des corps d'APDU.

4.3.2 Codages des data types

Les data types suivants, utilisés dans les APDU, sont représentés dans le Tableau 2. D'autres data types peuvent être acheminés comme données d'utilisateur.

Tableau 2 – Data types

Data type	Longueur d'octets	Commentaire
Boolean	1	0 = Faux, Non nul = VRAI
Integer8	1	
Integer16	2	
Unsigned8	1	
Unsigned16	2	
Unsigned32	4	
VisibleString	-	La longueur de VisibleString est définie par la définition du champ.
OctetString	-	La longueur de OctetString est définie par la définition du champ.
Time value	8	

Pour tous les types de données, l'ordre des bits et des octets de transmission est tel que défini dans le Type 9. La valeur entière non signée pour chaque position de bits est représentée dans le Tableau 3.

NOTE L'codage qui résulte est le même que celui spécifié dans l'Appendice B de RFC 791 – Internet Protocol. L'ordre des bits et des octets spécifié dans RFC 791 Annexe B est l'ordonnancement des octets gros-boutiste (l'octet de poids fort pour les nombres entiers) et l'ordonnancement des bits au sein des octets est un bit de poids fort. Bien que la numérotation des bits soit le contraire de ce qui est spécifié dans FMS, les deux spécifient le bit d'ordre supérieur (le bit le plus à gauche) comme étant le bit de poids fort. Dans les descriptions suivantes, le bit 1 est le bit de poids fort dans chaque octet. La valeur entière non signée pour chaque position de bits est représentée dans le tableau ci-dessous.

Tableau 3 – Data types

Position de bits	Valeur hexadécimale	Valeur décimale
8	0x80	128
7	0x40	64
6	0x20	32
5	0x10	16
4	0x08	8
3	0x04	4
2	0x02	2
1	0x01	1

4.3.3 En-tête d'APDU

4.3.3.1 Format de l'en-tête d'APDU

L'en-tête est le même pour chaque APDU et a une longueur fixe.

Tous les champs d'en-tête sont obligatoires.

Le format de l'en-tête est représenté dans le Tableau 4.

Tableau 4 – Format de l'en-tête d'APDU

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Type 5 APDU Version	0	Unsigned8	1	Spécifie le numéro de Version du format d'APDU de Type 5. Le numéro de version des APDU de Type 5 spécifié dans le présent document est égal à 1. 1 = Version 1
Options	1	Unsigned8	1	Bit 8: 1= Numéro d'APDU présent dans la fin de trame Bit 7: 1= Invoke Id présent dans la fin de trame ce bit doit être établi pour toutes les sessions client / serveur) Bit 6: 1 = Time Stamp présent dans la fin de trame Bit 5: Réservé, mis à 0 Bit 4: 1= Extended Control Field présent dans la fin de trame Bits 3-1: Pad Length = Spécifie le nombre d'octets de remplissage qui sont insérés entre les données d'utilisateur et les champs de fin de trame. Chaque octet de remplissage est mis sur zéro binaire.
ASE Id And Confirmed Msg Type	2	Unsigned8	1	Bits 8–3: ASE Id – Utilisé pour spécifier l'ASE dont le service est en cours d'acheminement par l'APDU. Il est utilisé comme un nombre entier non signé de 6 bits avec les valeurs suivantes: 0 = Unused 1 = AR ASE 2 = SMK ASE 3 = Type 9 ASEs/Type 5 VFD ASE 4 = LAN Redundancy ASE 5-63 : Reserved Bits 1, 2: Confirmed Msg Type – Utilisé pour différencier des APDU de demande, de réponse et d'erreur. Ne s'utilise pas avec des services non confirmés. Ces bits sont interprétés comme un nombre entier non signé de 2 bits avec les valeurs suivantes : 0 = Request APDU 1 = Response APDU 2 = Error APDU 3 = Reserved
Service	3	Unsigned8	1	Spécifie le service au sein du protocole spécifié. Bit 8: Confirmed Flag (Confirmed=1) and (Unconfirmed=0). Lorsque le fanion est établi pour une APDU de demande, la réponse correspondante (soit une APDU de réponse, soit une APDU d'erreur) est toujours renvoyée après le traitement de la demande pour indiquer le résultat du traitement. Bits 7-1 Service Id du service (nombre entier non signé de 7 bits). La valeur utilisée pour chaque Service Id est spécifiée dans sa spécification des corps d'APDU ci-dessous. C'est le nombre entre parenthèses qui suit le nom du service dans l'en-tête pour chacun des services. Toutes les autres valeurs sont réservées.
FDA Address	4	Unsigned32	4	Le format et l'utilisation de FDA Address dépendent de l'ASE Id et du type de S-VFD Context. Le paragraphe suivant décrit ce champ.
APDU Length	8	Unsigned32	4	Spécifie le nombre d'octets contenus dans une APDU entière, y compris l'en-tête et la fin de trame.

4.3.3.2 Utilisation de l'adresse de FDA

4.3.3.2.1 Résumé sur l'utilisation de l'adresse de FDA

Le Paragraphe 4.3.3.2.1 spécifie les valeurs pour le champ FDA Address de l'en-tête. Voir le Tableau 5 pour plus de détails. Dans les descriptions suivantes, un appareil de Type 9 est un appareil avec une AL de Type 9 et une DLL dont l'adressage peut être représenté comme étant une adresse de DLL de Type 1. Un SMK de Type 9 est le SMK dans un appareil de Type 9.

Tableau 5 – Utilisation de l'adresse de FDA

ASE	VCR	Bits	Utilisation
AR	N/A	32-1	Spécifique au service
LAN Redundancy	N/A	32-1	N'est pas utilisé, mis à 0.
SMK	N/A	32-17	Link Id Mis à 0 si le SMK de destination est situé dans un appareil de Type 5, destinataire des APDU. S'il s'agit d'un appareil de liaison et le SMK de destination est situé dans un appareil de Type 9 attaché à l'appareil de liaison, le champ FDA Address contient alors l'identificateur de liaison (Link id) utilisé pour accéder à l'appareil de Type 9.
		16-1	Node.Selector La valeur du nœud est égale aux 8 bits d'ordre supérieur et la valeur du sélecteur est égale aux 8 bits d'ordre inférieur lorsque considérées comme une valeur non signée de 16 bits. Pour identifier un SMK de Type 5 (quand Link Id est égal à zéro), la valeur du nœud est mise à 0 et la valeur du sélecteur est mise à 2. Il s'agit d'une adresse node.selector de groupe ou individuelle de Type 9 utilisée pour identifier un SMK de Type 9 (quand Link Id n'est pas égal à zéro). Voir la description d'APDU de Find Tag Query qui donne une présentation détaillée de l'utilisation de FDA Address.
Type 9	Client/Server	32-17	Link Id Si le serveur S-VFD est situé dans un appareil de Type 5, il s'agit alors de 0. Si c'est un appareil de liaison et le VFD de destination se situe dans un appareil de Type 9 attaché à l'appareil de liaison, il s'agit alors de Link Id de la liaison à laquelle l'appareil de Type 9 est attaché.
		16-1	Type 5 Selector Demande initiée: L'utilisation de ce champ dépend de Connect Option sélectionnée pour la demande. Voir la description d'APDU initiée pour les valeurs utilisées. Réponse initiée: AE de FAL de Type 5 retourne le VCR Id du VCR initié. Tous les autres: Le Sélecteur renvoyé dans la réponse initiée.
Type 9	Publisher / Subscriber	32-17	Link Id Si le VFD éditeur est situé dans un appareil de Type 5 émettant des APDU, il est alors combiné avec les bits du Sélecteur pour former une adresse directe DLcep de 32 bits. Si c'est un appareil de liaison, et l'éditeur se situe dans un appareil de Type 9 attaché à l'appareil de liaison, il s'agit alors de Link Id de la liaison à laquelle l'appareil d'édition est attaché.
		16-1	Dlcep Selector Si l'éditeur est situé dans un appareil de Type 5 émettant des APDU, il s'agit alors d'une partie de l'adresse directe Dlcep de 32 bits attribuée aux données éditées. Si c'est un appareil de liaison et l'éditeur se trouve dans un appareil de Type 9 attaché à l'appareil de liaison, il s'agit alors de Dlcep Selector du tampon d'éditeur.
Type 9	Report	32-	Link Id

ASE	VCR	Bits	Utilisation
	Distribution	17	Si VFD source d'alertes est situé dans un appareil de Type 5 émettant des APDU, il est alors combiné avec les bits du Sélecteur pour former une adresse directe de DLSAP de 32 bits. Si c'est un appareil de liaison, et la source d'alertes se trouve dans un appareil de Type 9 attaché à l'appareil de liaison, il s'agit alors de Link Id de la liaison à laquelle l'appareil d'alertes est attaché.
		16-1	S-VFD Context Id ou Dlsap Selector Si VFD source d'alertes est situé dans un appareil de Type 5 émettant des APDU, il s'agit alors d'une partie de l'adresse directe Dlsap de 32 bits attribuée à Report Source VFD. Si c'est un appareil de liaison, et l'éditeur se situe dans un appareil de Type 9 attaché à l'appareil de liaison, il s'agit alors du Sélecteur Dlsap de la source d'alertes.
NOTE Link Id, Node, DLCEP Selector et DLSAP Selectors sont tous des composants de l'adresse de DLL de Type 1.			

4.3.3.2.2 APDU envoyée par un point d'extrémité VCR client

Le Tableau 6 représente l'utilisation du champ d'en-tête de FDA Address dans des APDU envoyées par un point d'extrémité VCR client.

Tableau 6 – APDU du champ d'en-tête de FDA Address envoyées par un point d'extrémité VCR client

Type de message	AP de destination	Adresse de FDA	
		Bits	Valeur
APDU de demande initiée de Type 5 Connect Option 1	Non-MIB VFD d'un appareil de Type 5	Bits 32-17 Bits 16-1	0 (appareil local de Type 5) Generic Selector renvoyé par Find Tag Query ou obtenu par d'autres moyens
	Non-MIB VFD d'un appareil de Type 9 connecté à l'appareil de liaison	Bits 32-17 Bits 16-1	Link Id d'une Liaison de Type 9 Client Type 9 VCR Index
APDU de demande initiée de Type 5 Connect Option 2	MIB VFD d'un appareil de Type 5	Bits 32-17 Bits 16-1	0 (appareil local de Type 5) 0.0
	MIB VFD d'un appareil de liaison de Type 9 Interface de liaison	Bits 32-17 Bits 16-1	Link Id d'une Liaison de Type 9 0.0
	MIB VFD d'un appareil de Type 9 connecté à l'appareil de liaison	Bits 32-17 Bits 16-1	Link Id d'une Liaison de Type 9 Node.0
Tous les autres messages de demande	MIB VFD d'un appareil de Type 5	Bits 32-1	FDA Address renvoyée dans Un message de la réponse initiée
	Non-MIB VFD	Bits 32-1	FDA Address renvoyée dans Un message de la réponse initiée
	MIB VFD d'un appareil de liaison de Type 9 Interface de liaison	Bits 32-1	FDA Address renvoyée dans Un message de la réponse initiée
	MIB VFD d'un appareil de Type 9	Bits 32-1	FDA Address renvoyée dans

Type de message	AP de destination	Adresse de FDA	
		Bits	Valeur
	connecté à l'appareil de liaison		Un message de la réponse initiée
	Non-MIB AP d'un appareil de Type 9	Bits 32-1	FDA Address renvoyée dans
	connecté à l'appareil de liaison		Un message de la réponse initiée

4.3.3.2.3 APDU envoyées par un point d'extrémité VCR AR serveur

Le Tableau 7 représente l'utilisation du champ d'en-tête de FDA Address dans les APDU envoyées par un point d'extrémité VCR serveur.

Tableau 7 – APDU du champ d'en-tête de l'adresse de FDA envoyées par un point d'extrémité VCR serveur

Type de message	AP de destination	Adresse de FDA	
		Bits	Valeur
APDU de réponse initiée de Type 5 Connect Option 1	Non-MIB VFD d'un appareil de Type 5	Bits 32-17 Bits 16-1	0 (appareil local de Type 5) Type 5 VCR Id[1]
	Non-MIB VFD d'un appareil de Type 9 connecté à l'appareil de liaison	Bits 32-17 Bits 16-1	Link Id d'une Liaison de Type 9 Type 5 VCR Id[1]
APDU de réponse initiée de Type 5 Connect Option 2	MIB VFD d'un appareil de Type 5	Bits 32-17 Bits 16-1	0 (appareil local de Type 5) Type 5 VCR Id[1]
	MIB VFD d'un appareil de liaison de Type 9 Interface de liaison	Bits 32-17 Bits 16-1	Link Id d'une Interface de Liaison de Type 9 Type 5 VCR Id[1]
	MIB VFD d'un appareil de Type 9 connecté à l'appareil de liaison	Bits 32-17 Bits 16-1	Link Id d'une Liaison de Type 9 Type 5 VCR Id[1]
Tous les autres messages de demande	MIB VFD d'un appareil de Type 5	Bits 32-1	La même que message de demande La même que message de demande
	Non MIB VFD d'un appareil de Type 5	Bits 32-1	La même que message de demande La même que message de demande
	MIB VFD d'un appareil de liaison de Type 9 Liaison	Bits 32-1	La même que message de demande La même que message de demande
	MIB VFD d'un appareil de Type 9 connecté à l'appareil de liaison	Bits 32-1	La même que message de demande La même que message de demande
	Non-MIB VFD d'un appareil de Type 9 connecté à l'appareil de liaison	Bits 32-1	La même que message de demande La même que message de demande
NOTE La valeur du sélecteur (17-32 bits) renvoyée est Index VCR de Type 5 du VCR créé de façon dynamique.			

4.3.3.2.4 APDU envoyées par un point d'extrémité VCR éditeur

Le Tableau 8 représente l'utilisation de FDA Address dans les messages envoyés par un point d'extrémité VCR éditeur.

Tableau 8 – APDU du champ d'en-tête de l'adresse de FDA envoyées par un point d'extrémité VCR éditeur

Type de message	AP éditeur	Adresse de FDA	
		Bits	Valeur
Message de demande de Type 9	AP dans l'appareil de Type 5	Bits 32-1	Adresse directe de Type 9 de 32 bits attribuée au tampon de données éditées
	VFD d'un appareil de Type 9 connecté par une	Bits 32-17	Link Id d'une Liaison de Type 9
	Liaison de Type 9 d'un appareil de liaison	Bits 16-1	Node.Selector Dlcep du tampon de données éditées

4.3.3.2.5 Messages envoyés par un point d'extrémité VCR source d'alertes

Le Tableau 9 représente l'utilisation de FDA Address dans les APDU envoyées par un point d'extrémité VCR source d'alertes.

Tableau 9 – APDU du champ d'en-tête de l'adresse de FDA envoyées par un point d'extrémité VCR source d'alertes

Type de message	AP de source d'alertes	Adresse de FDA	
		Bits	Valeur
Message de demande de Type 9	AP dans l'appareil de Type 5	Bits 32-1	Adresse directe de Type 9 de 32 bits attribuée à VFD de source d'alertes
	VFD d'un appareil de Type 9 connecté par une	Bits 32-17	Link Id d'une Liaison de Type 9
	Liaison de Type 9 d'un appareil de liaison	Bits 16-1	Node.Selector Dlsap de la source d'alertes

4.3.4 Fin de trame

Les Sessions de FDA déterminent les champs de fin de trame qui sont définis pour les APDU de Type 9 qu'elles acheminent. L'utilisation des champs de fin de trame par SMK, ASE et LAN Redundancy APDU est spécifique au service. Voir les définitions de SMK, ASE et LAN Redundancy APDU pour leur utilisation des champs de fin de trame.

La présence des champs de fin de trame est négociée pour les sessions Client / Serveur et est configurée pour les sessions Éditeur / Abonné et Diffusion de Rapports. Leur présence est indiquée dans le champ "Message Header Options" de chacun des messages. Les champs présents sont placés après les données d'utilisateur dans l'ordre représenté dans le Tableau 10.

Tableau 10 – Format de l'en-tête d'APDU

Nom du champ	Ordre	Type de données	Longueur d'octets	Description
APDU Number	1	Unsigned32	4	Basé sur 0, augmenté de façon monotone par 1, nombre d'APDU au sein d'une session spécifique.
Invoke Id	2	Unsigned32	4	Identificateur de demande/réponse utilisé pour que les réponses correspondent aux demandes. C'est exigé pour tous les échanges de demande / réponse. Invoke Id sont uniques dans le contexte d'une AR.
Time Stamp	3	OctetString	8	Temps systémique où l'émetteur a créé une APDU (dans le format d'une valeur temporelle). Il peut être utilisé par tous les destinataires pour calculer le temps du transfert à sens unique en provenance de l'émetteur.
Extended Control Field	4	Unsigned32	4	Réservé pour une future utilisation.

4.3.5 Corps d'APDU

4.3.5.1 Généralités

Chaque service possède un ensemble de corps d'APDU qui sont acheminés sous format fixe dans chaque corps d'APDU. La longueur des formats de corps d'APDU peut varier seulement si le dernier champ du corps d'APDU se répète ou si sa longueur reste variable. Les champs de longueur variable sont pris en charge seulement en tant que le dernier champ d'un corps d'APDU (avant la fin de trame d'APDU) pour que leur décalage dès le début du corps d'APDU soit constant.

Le chiffre entre parenthèses qui suit le nom du service dans chaque en-tête des paragraphes ci-dessous représente la valeur utilisée pour le champ Service Id dans l'en-tête d'APDU de Type 5.

4.3.5.2 Services de ASE de la Session FDA

4.3.5.2.1 Service de Session Ouverte (service confirmé Id = 1)

4.3.5.2.1.1 Vue d'ensemble

Ce service est utilisé pour ouvrir une AR Client / Service entre un point d'extrémité d'AR client et un point d'extrémité d'AR serveur. Une paire d'adresses de source et de destination identifie chaque point d'extrémité d'AR.

Le champ Version de l'en-tête d'APDU de Type 5 dans le message de demande indique la version de l'Agent de FDA qui est en cours de demande d'AR. Le répondeur peut négocier la version à la baisse.

Le champ Options de l'en-tête d'APDU de Type 5 dans le message de demande indique les options qui sont en cours de demande d'AR. Invoke Id est toujours établi. Le répondeur peut refuser toutes les autres options. Le retour de la valeur 0 vers la position de bits indique le refus.

Si le numéro du message est demandé comme option, le champ de fin de trame contient la valeur initiale à utiliser par les deux points d'extrémité de la session. Cette valeur est renvoyée dans le message de réponse. Si le demandeur ne reçoit pas la réponse à la demande et choisit de renvoyer la demande, il augmente le numéro du message dans la fin de trame tout en maintenant le même identifiant d'invocation par rapport à l'original. Cela déclenche la livraison de la demande au point d'extrémité d'AR à condition que la première

demande ait été reçue avec le retour d'une réponse positive. Si la demande initiale n'a pas été reçue ou si elle a généré une réponse négative en retour (sans être reçue), la nouvelle demande est alors traitée comme une nouvelle demande, car il n'existe pas de point d'extrémité d'AR pour le traitement du message.

L'adresse de FDA dans l'en-tête d'APDU n'est pas utilisée et est mise à 0.

Si le serveur reçoit un message de demande de Session Ouverte pour une session sur laquelle il a déjà envoyé un message de réponse de Session Ouverte, il ferme la session. Sinon, il s'agit des applications suivantes.

Le champ Version de l'en-tête de Messages de FDA indique dans le message de demande la version de FDA qui est en cours de demande de session. Le répondeur peut négocier la version à la baisse.

Le message de demande de Session Ouverte achemine des attributs de session pour la validation et l'utilisation par le répondeur. Si le répondeur est capable de maintenir certains d'entre eux, il est autorisé de les négocier comme défini ci-dessous dans le tableau des Paramètres de Messages de Demande. Si le demandeur n'est pas prêt à fonctionner avec les attributs négociés et renvoyés par le répondeur, il peut ne pas ouvrir la session. Il peut ensuite renvoyer une demande de Session Ouverte avec différents attributs de la session, sinon, en utilisant TCP, fermer la connexion TCP associée pour pouvoir fermer la session appropriée.

Le champ Options en-tête des messages dans le message de demande indique les options qui sont en cours de demande pour la session. Invoke Id est toujours établi. Le répondeur peut refuser toutes les autres options. L'option de retour de la valeur 0 vers la position de bits indique le refus.

Si le numéro de message est demandé comme option, le champ de fin de trame contient la valeur initiale à utiliser par les deux points d'extrémité de la session. Cette valeur est renvoyée dans le message de réponse. Si le demandeur ne reçoit pas la réponse à la demande et choisit de renvoyer la demande, il augmente le numéro du message dans la fin de trame tout en maintenant le même identifiant d'invocation par rapport à l'original. Cela génère la livraison de la demande au point d'extrémité de la session créée à condition que la première demande ait été reçue avec le retour d'une réponse positive. Cela cause la fermeture de la session. Si la demande initiale n'a pas été reçue ou si elle a provoqué une réponse négative en retour (sans être reçue), la nouvelle demande est alors traitée comme une nouvelle demande, car il n'existe pas de point d'extrémité de la session pour traiter le message.

Pour l'option PadLength, le demandeur spécifie la limite d'alignement pour le message. Cela est effectué en mettant PadLength sur un nombre maximal d'octets de remplissage qui peuvent être insérés, mais également en remplissant ce message de façon appropriée. Si cette valeur nécessite le remplissage de 4 ou 8 octets, alors que le remplissage n'est pas pris en charge par le destinataire, il peut rejeter la demande à l'aide des "options non valides" ou l'accepter et retourner "000" dans ce champ.

000 = Pas de remplissage

011 = remplissage jusqu'à une limite de 4 octets. Le nombre maximal d'octets de remplissage, qui peuvent être insérés, est égal à 3. Trois octets de remplissage sont insérés dans ce message.

111 = remplissage jusqu'à une limite de 8 octets Le nombre maximal d'octets de remplissage, qui peuvent être insérés, est égal à 7. Sept octets de remplissage sont insérés dans ce message.

Le serveur reçoit des demandes à une adresse de point d'extrémité générique et crée un nouveau point d'extrémité à une adresse qui n'a pas été utilisée précédemment afin de traiter chaque demande et de retourner la réponse. Le nouveau point d'extrémité est utilisé pour toutes les APDU ultérieures envoyées et reçues sur l'AR.

Si une demande est reçue avec un paramètre non valide ou avec des valeurs non prises en charge, une réponse négative est renvoyée avec une classe d'erreur "service" et un code d'erreur "parameter-inconsistent" pour le paramètre incorrect. Cette réponse négative utilise également la valeur du 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 l'ouverture de la session. Pour acheminer ces valeurs, les 16 octets de la description supplémentaire sont interprétés comme des nombres entiers 4 Unsigned32 au lieu de VisibleString. L'emplacement de chaque valeur du paramètre pour les 16 octets est spécifié dans les descriptions ci-dessous.

Si le répondeur ne dispose pas d'Index de Session disponible pour la nouvelle session, une réponse négative est alors renvoyée avec une classe d'erreur = "resource" et un code d'erreur = "max sessions exceeded".

Si le répondeur ne dispose pas de ressources pour prendre en charge la nouvelle session, une réponse négative est alors renvoyée avec une classe d'erreur = "resource" et un code d'erreur "object creation failure" ou avec un autre code d'erreur approprié au sein d'une classe d'erreur "resource".

Un échange réussi des messages de demande et de réponse de Session Ouverte donne lieu à une session qui peut être utilisée pour envoyer des messages de demande et de réponse de FMS. Dans le cas d'utilisation de UDP, le serveur renvoie la réponse en provenance d'un port UDP qui n'a pas été utilisé précédemment. Ce port est alors utilisé pour envoyer et recevoir tous les messages ultérieurs pour la session.

Des AR ouvertes pour l'utilisation de la configuration MIB VFD sont dénommées comme des AR de configuration. Une seule configuration d'AR est autorisée en une seule fois. Si une demande est reçue pour ouvrir une AR de configuration alors qu'il y en a déjà une ouverte ou s'il existe un VCR de Type 9 pour tout MIB VFD (Type 5 ou Type 9) avec les services de mise à jour pris en charge et déjà ouverts, la configuration d'AR est refusée. La ASE d'AR retourne une réponse négative avec une classe d'erreur = "service" et un code d'erreur = "config-acces-already-open".

L'ouverture des AR éditeur / abonné et diffusion de rapports est locale et ne résulte pas dans l'échange des APDU du Service de Session Ouverte. Par conséquent, les paramètres d'APDU de demande de Session Ouverte pour ces types d'AR sont préconfigurés au lieu d'être négociés.

4.3.5.2.1.2 Paramètres d'APDU de demande

Le Tableau 11 liste les paramètres s'APDU de demande.

Tableau 11 – Paramètres d'APDU de demande

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
AR Index	0	Unsigned32	4	Ce paramètre contient l'identificateur numérique du point d'extrémité AR, s'il en existe un. Sinon, on utilise 0. Pour la réponse, ce paramètre contient l'identificateur numérique d'une toute nouvelle AR.
Max Buffer Size	4	Unsigned32	4	Ce paramètre définit la longueur maximale du tampon en octets (le tampon peut contenir des APDU enchaînées de Type 9) qui peuvent être envoyées ou reçues par l'émetteur de cette APDU sur cette AR. Ce paramètre peut être négocié à la baisse (mais pas à la montée) par le répondeur à l'aide du paramètre Max Buffer Size. Dans le cas d'échec de la négociation pour un paramètre, une valeur acceptable pour ce paramètre est renvoyée dans le champ de la description supplémentaire comme Unsigned32 au décalage 0. Si la valeur demandée est prise en charge, mais un autre paramètre n'a pas réussi la négociation, la valeur demandée est renvoyée au décalage de 0.
Max Message Length	8	Unsigned32	4	Ce paramètre définit la longueur maximale en octets des APDU à envoyer sur cette AR par l'émetteur de cette APDU. Il est utilisé par le point d'extrémité destinataire AR en tant que son attribut Max Message Length. Ce paramètre ne peut pas être négocié. Le serveur peut rejeter la valeur fournie par le client, s'il ne peut pas la maintenir à l'aide de la classe d'erreur = "access" et le code d'erreur = "unsupported max APDU length". Dans le cas d'échec de la négociation pour un paramètre, une valeur acceptable pour ce paramètre est renvoyée dans le champ de la description supplémentaire comme Unsigned32 au décalage 4. Si la valeur demandée est prise en charge, mais un autre paramètre n'a pas réussi la négociation, la valeur demandée est renvoyée au décalage de 4.
Reserved	12	Unsigned8	1	Ne s'utilise pas, mis à 0
Configuration Use	13	Unsigned8	1	0 = Configuration n'est pas autorisée 1 = Configuration est autorisée
Inactivity Close Time	14	Unsigned16	2	Ce paramètre identifie le temps (en secondes) pendant lequel l'AR reste ouverte sans recevoir une APDU. Après avoir été inactif sur l'AR pendant cette période de temps, l'AREP se ferme avec toute activité VCR associée à l'AREP. Le répondeur peut négocier cette valeur à la baisse. La valeur 0 n'est pas autorisée. Dans le cas d'échec de la négociation pour un paramètre, une valeur acceptable pour ce paramètre est renvoyée dans le champ de la description supplémentaire comme Unsigned32 au décalage 8. Si la valeur demandée est prise en charge, mais un autre paramètre n'a pas réussi la négociation, la valeur demandée est renvoyée au décalage de 8.

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Transmit Delay Time	16	Unsigned32	4	Ce paramètre est utilisé pour établir l'attribut Transmit Delay Time dans l'AREP récepteur. Sa valeur ne peut pas être négociée. Elle est exprimée en millisecondes. Dans le cas d'échec de la négociation pour un paramètre, une valeur acceptable pour ce paramètre est renvoyée dans le champ de la description supplémentaire comme Unsigned32 au décalage 12. Si la valeur demandée est prise en charge, mais un autre paramètre n'a pas réussi la négociation, la valeur demandée est renvoyée au décalage de 12.
SMK PD Tag	20	VisibleString	32	SMK PD Tag du serveur. Si le SMK PD Tag dans le message de demande ne correspond pas au PD Tag du serveur, le serveur rejette la demande avec la classe d'erreur 'access" et le code d'erreur "object-access-denied".

4.3.5.2.1.3 Paramètres d'APDU de réponse

Les mêmes que pour APDU de demande

4.3.5.2.1.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.2.2 Idle (service confirmé Id = 3)

4.3.5.2.2.1 Vue d'ensemble

Cette APDU est utilisée pour informer le destinataire de la présence de l'émetteur.

4.3.5.2.2.2 Paramètres d'APDU de demande

Néant.

4.3.5.2.2.3 Paramètres d'APDU de réponse

Néant.

4.3.5.3 Services de ASE de SMK

4.3.5.3.1 SM find tag query (service non confirmé Id = 1)

4.3.5.3.1.1 Vue d'ensemble

L'adresse de réseau utilisée pour envoyer le message FIND_TAG_QUERY et l'adresse de FDA dans l'en-tête d'APDU de Type indiquent ensemble les SMK devant recevoir et traiter le message.

Le message Find Tag Query peut être envoyé à un ou plusieurs SMK dans des appareils de Type 5 et/ou de Type 9. Une fois envoyé à un appareil de liaison, le SMK de Type 5 de l'appareil de liaison répond, si l'appareil de liaison contient l'objet interrogé. Les SMK d'interface de Type 9 dans l'appareil de liaison ne sont utilisés que pour faire suivre l'interrogation vers les liaisons de Type 9. Ils ne répondent jamais aux interrogations.

Bien que ce service soit non confirmé, il utilise Invoke Id dans la fin de trame de messages pour qu'il corresponde au message Find Tag Reply de SM renvoyé.

Le Tableau 12 décrit la diffusion qui peut être demandée pour ce service. Les adresses de FDA dans le tableau sont représentées en hexadécimales.

Tableau 12 – Valeurs de l'adresse de FDA de SMK

Signification de l'adresse de FDA	Adresse de FDA		Adresse de destination	Diffusion
	LLLL	NN.SS		
SMK de l'appareil de Type 5	00 00	00.02	SMK Multicast	Tout SMK de Type 5
			Individuel	SMK de Type 5 individuel
Adresse de groupe de la liaison locale de SMK de Type 5 et de Type 9	00 00	01.09[1]	SMK Multicast	Tout SMK de Type 5 et tout SMK de Type 9
			Individuel	SMK de LD individuel et tout SMK de Type 9 accessible à travers LD Les SMK des interfaces de Type 9 des appareils de liaison ne sont pas inclus.
Adresse de groupe de la liaison locale de SMK de Type 9	LL LL	01.09[1]	SMK Multicast	Tout SMK de Type 9 sur LL LL de liaison de Type 9
			Individuel	Tout SMK de Type 9 sur LL LL de liaison de Type 9 accessible à travers LD. Les SMK des interfaces LL LL de Type 9 des appareils de liaison ne sont pas inclus.
Adresse de Type 9 individuelle	LL LL	NN.02	Individuel	SMK individuel de Type 9 accessible à travers LD

NOTE Spécifié dans la CEI 61158-1 comme l'adresse spécifique de liaison de Type 1 pour "les SMAE de toutes les DLE sur la liaison".

4.3.5.3.1.2 Paramètres d'APDU de demande

Le Tableau 13 représente les valeurs de l'adresse de FDA de SMK.

Tableau 13 – Valeurs de l'adresse de FDA de SMK

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Query Type	0	Unsigned8	1	Indique le type d'interrogation : 0 = PD Tag query pour un appareil primaire ou non redondant 1 = VFD tag query 2 = Function-Block tag query 3 = Element Id query 4 = PD Tag/VFD Reference query 5 = Device Index query 6 = PD Tag query pour un appareil secondaire ou le membre d'un ensemble redondant Toutes les autres valeurs sont réservées.
Reserved	1	OctetString	3	Réservé, mis à 0
Element Id Or VFD Reference	4	Unsigned32	4	Paramètre de Service Mis à 0 binaire quand il n'est pas utilisé.
PD Tag Or Object Tag	8	VisibleString	32	Paramètre de Service Mis à 0 binaire quand il n'est pas utilisé.
VFD Tag	40	VisibleString	32	Paramètre de Service Mis à 0 binaire quand il n'est pas utilisé.

4.3.5.3.2 SM find tag reply (service non confirmé Id = 2)

4.3.5.3.2.1 Vue d'ensemble

Ce message est utilisé pour répondre au message de demande de SM Find Tag

Bien que ce service soit non confirmé, il utilise Invoke Id dans la fin de trame d'APDU de Type 5 pour qu'il corresponde au message de SM Find Tag Query.

4.3.5.3.2.2 Paramètres d'APDU de demande

Le Tableau 14 représente les paramètres d'APDU de demande.

Tableau 14 – Paramètres d'APDU de demande

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Query Type	0	Unsigned8	1	Indique le type d'interrogation : 0 = PD Tag query pour un appareil primaire ou non redondant 1 = VFD tag query 2 = Function-Block tag query 3 = Element Id query 4 = PD Tag/VFD Reference query 5 = Device Index query 6 = PD Tag query pour un appareil secondaire ou le membre d'un ensemble redondant Toutes les autres valeurs sont réservées.
Type 9 Node Address	1	Unsigned8	1	Paramètre de Service Mis à 0 si la Réponse est pour un appareil de Type 5.
Type 9 Link Id	2	Unsigned16	2	Paramètre de Service Mis à 0 si la Réponse est pour un appareil de Type 5.
VFD Reference	4	Unsigned32	4	Paramètre de Service
Queried Object Numeric Id	8	Unsigned32	4	Paramètre de Service
Queried Object Network Address	12	OctetString	16	Paramètre de Service
Queried Object OD Version Number	28	Unsigned32	4	Paramètre de Service
Queried Object Device ID	32	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Queried Object PD Tag	64	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Duplicate Detection State	96	Unsigned8	1	Paramètre de Service Encodé comme suit. Bits 8–3: Réservé, mis à 0 Bit 2: 1 = PD Tag dupliqué détecté 0 = PD Tag dupliqué non détecté Bit 1: 1 = Device Index dupliqué

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
				détecté 0 = Device Index dupliqué non détecté
Reserved	97	Unsigned8	1	Réservé, mis à 0
Number in List of FDA Addresses	98	Unsigned16	2	Ce paramètre indique combien Sélecteurs de l'adresse de FDA sont contenus dans la Liste de paramètres des Sélecteurs de l'adresse de FDA. La valeur de cet attribut est égale à 0 si le Type l'interrogation est "physical-device-query".
List of FDA Addresses	100	Unsigned16	2x number in list	Paramètre de Service Chaque Sélecteur représente Unsigned32 qui suit immédiatement son prédécesseur dans la série (pas d'espaces ou de remplissage entre eux).

4.3.5.3.3 SM Identify (service confirmé Id = 3)

4.3.5.3.3.1 Vue d'ensemble

Ce service est utilisé pour demander l'identité d'un seul appareil de Type 5 ou de Type 9 ou d'un groupe d'appareils de Type 5.

Le Tableau 15 spécifie les adresses de réseau et les adresses de FDA utilisées avec le message de demande de SM Identity.

Tableau 15 – Valeurs d'adresse de FDA de SMK pour SM identity

Identification de	Adresse de FDA LLLL.NN.SS	Adresse de réseau	Message de demande livré à
Appareil de SMK de Type 5 y compris	00 00.00.02	SM Multicast	Tout SMK de Type 5
Live List Version Number List[a]		Individuel	SMK de Type 5 individuel
Liste des numéros de la version de nœud de l'appareil de liaison pour une Interface spécifique de Type 9	LL LL.00.02	SM Multicast	Tout SMK de Type 5 de l'appareil de liaison – réponse de l'un avec Link Id demandé
	LL LL.00.02	Individuel	SMK de Type 5 de l'appareil de liaison
SMK d'un appareil de Type 9	LL LL.NN.02	Individuel	appareil de liaison de Type 5 -> SMK de Type 9

[a] The Live List Version Number List est renvoyée par les SMK d'appareil de liaison.

4.3.5.3.3.2 Paramètres d'APDU de demande

Néant.

4.3.5.3.3.3 Paramètres d'APDU de réponse

Les mêmes que les paramètres d'APDU de demande de SM Device Annunciation

4.3.5.3.3.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.3.4 SMK clear adress (service confirmé Id = 12)

4.3.5.3.4.1 Vue d'ensemble

Toujours confirmé, ce service utilise Invoke Id dans la fin de trame de message.

Le Tableau 16 spécifie l'adresse de réseau et l'adresse de FDA utilisées avec les APDU de réponse et de demande de SM Clear Address.

Tableau 16 – Valeurs de l'adresse de FDA de SMK pour les APDU de demande de SMK set assignment info

Utilisation	Adresse de FDA LLLL.NN.SS	Adresse de réseau	Trajet de demande
Type 5 Clear Address	00 00.00.02	Individuel	Demandeur de Type 5 -> SMK de Type 5
Type 9 Clear Address	LL LL.NN.02	Individuel	Demandeur de Type 5 -> Agent de FDA LD -> SMK de Type 9

4.3.5.3.4.2 Paramètres d'APDU de demande

Le Tableau 17 spécifie les paramètres d'APDU de demande.

Tableau 17 – Paramètres d'APDU de demande de SMK clear address

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Device Id	0	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Physical Device Tag	32	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
InterfaceToClear	64	Unsigned8	1	Paramètre de Service Encodé comme suit : 0 = Interface A 1 = Interface B 255 = Toute interface
Reserved	65	OctetString	3	Réservé, mis à 0

4.3.5.3.4.3 Paramètres d'APDU de réponse

Néant.

4.3.5.3.4.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.3.5 SMK set assignment info (service confirmé Id = 14)

4.3.5.3.5.1 Vue d'ensemble

Toujours confirmé, ce service utilise Invoke Id dans la fin de trame de message.

Le Tableau 18 spécifie l'adresse de réseau et l'adresse de FDA utilisées avec les APDU de réponse et de demande de SM Set Assignment Info.

Tableau 18 – Valeurs de l'adresse de FDA de SMK pour les APDU de demande de SMK set assignment info

Utilisation	Adresse de FDA LLLL.NN.SS	Adresse de réseau	Trajet de demande
Set Assignment Info de Type 5	00 00.00.02	Individuel	Demandeur de Type 5 -> SMK de Type 5
Set Assignment Info de Type 9	LL LL.NN.02	Individuel	Demandeur de Type 5 -> Agent de FDA LD -> SMK de Type 9

4.3.5.3.5.2 Paramètres d'APDU de demande

Le Tableau 19 spécifie les paramètres d'APDU de demande.

Tableau 19 – Paramètres d'APDU de demande de SMK set assignment

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Device Id	0	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Physical Device Tag	32	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Type 9 New Address	64	Unsigned8	1	Paramètre de Service Le champ contient la nouvelle adresse de Type 9 pour l'appareil de Type 9 identifié dans FDA Address de l'en-tête d'APDU de Type 5. La valeur de ce champ est égale à 0, si FDA Address n'identifie pas un appareil de Type 9 ou cette APDU n'est pas en cours d'utilisation pour changer l'adresse de l'appareil de Type 9.
Device Redundancy State	65	Unsigned8	2	Paramètre de Service Inutilisé et mis à 0 si FDA Address identifie un appareil de Type 9. La valeur 0 indique que l'appareil ne participe pas dans la redondance de l'appareil. S'il y participe, la valeur d'une paire de bits 2 et 1 est non nulle. La valeur d'une paire de bits 4 et 3 est non nulle toutes les fois que les bits 2 et 1 indiquent "External Switchover Control Device". Quand une paire de bits 1 et 2 indique "External Switchover control Device", les bits 3 et 4 doivent être mis à "2" pour indiquer que la première en cours doit être secondaire, et l'une des secondaires doit devenir première. Bit 8 = Réservé, mis à 0 Bit 7 = Réservé, mis à 0 Bit 6 = Réservé, mis à 0 Bit 5 = Réservé, mis à 0 Bits 4 et 3 -External Switchover Control Device Redundancy Role 0 = Inutilisé 1 = Primaire

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
				2 = Secondaire Bits 2 et 1 – Assigned Device Redundancy Type 0 = Non-redondant 1 = External Switchover Control 2 = External Switchover Control
LAN Redundancy Socket Address	66	Unsigned16	2	Paramètre de Service Inutilisé et mis à 0, si FDA Address identifie un appareil de Type 9. Mis au port Reserved LAN Redundancy, si FDA Address identifie un appareil de Type 5.
Annunciation Repeat Time	68	Unsigned32	4	Paramètre de Service Inutilisé et mis à 0, si FDA Address identifie un appareil de Type 9.
Device Index	72	Unsigned16	2	Paramètre de Service Encodé comme suit. 0 = Index non attribué 1 à Max Device Index = Device Index
Max Device Index	74	Unsigned16	2	Paramètre de Service Inutilisé et mis à 0, si FDA Address identifie un appareil de Type 9.
Operational Network Address	76	OctetString	16	Paramètre de Service Inutilisé et mis à 0, si FDA Address identifie un appareil de Type "X".
Reserved	92	OctetString	3	Réservé, mis à 0
Clear Duplicate Detection State	95	Unsigned8	2	Paramètre de Service Inutilisé et mis à 0, si FDA Address identifie un appareil de Type 9. Encodé comme suit : Bits 8–3: Réservé, mis à 0 Bit 2: 1 = Do not Duplicate PD Tag Détecté 0 = Clear Duplicate PD Tag Détecté Bit 1: 1 = Do not clear Duplicate Device Index Détecté 0 = Clear Duplicate Device Index Détecté

4.3.5.3.5.3 Paramètres d'APDU de réponse

Le Tableau 20 représente les paramètres d'APDU de réponse.

Tableau 20 – Paramètres d'APDU de réponse de SMK set assignment info

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Reserved	0	Unsigned16	2	Réservé, mis à 0
Max Device Index	2	Unsigned16	2	Paramètre de Service Inutilisé et mis à 0, si FDA Address identifie un appareil de Type 9.
Annunciation Repeat Time	4	Unsigned32	4	Paramètre de Service Inutilisé et mis à 0, si FDA Address identifie un appareil de Type 9.

4.3.5.3.5.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.3.6 SMK clear assignment info (service confirmé Id = 15)

4.3.5.3.6.1 Vue d'ensemble

Toujours confirmé, ce service utilise Invoke Id dans la fin de trame de message.

Le Tableau 21 spécifie l'adresse de réseau et l'adresse de FDA utilisées avec les APDU de réponse et de demande de SM Clear Assignment Info.

Tableau 21 – Valeurs de l'adresse de FDA de SMK pour les APDU de SMK device clear assignment info

Utilisation	Adresse de FDA LLLL.NN.SS	Adresse de réseau	Trajet de demande
Clear Assignment Info de Type 5	00 00.00.02	Individuel	Demandeur de Type 5 -> SMK de Type 5
Clear Assignment Info de Type 9	LL LL.NN.02	Individuel	Demandeur de Type 5 -> Agent de FDA LD -> SMK de Type 9

4.3.5.3.6.2 Paramètres d'APDU de demande

Le Tableau 22 définit les paramètres d'APDU de demande.

Tableau 22 – Paramètres d'APDU de demande de SMK clear assignment info

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Device Id	0	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Physical Device Tag	32	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.

4.3.5.3.6.3 Paramètres d'APDU de réponse

Néant.

4.3.5.3.6.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.3.7 SMK device annunciation (service non confirmé Id = 16)

4.3.5.3.7.1 Vue d'ensemble

Le format du corps d'APDU de demande défini ci-dessous est utilisé pour le service SM Device Annunciation et également pour le service SM Identity.

Ce service est toujours non confirmé. Aucun champ de fin de trame d'APDU n'est utilisé.

Le Tableau 23 spécifie l'adresse de réseau et l'adresse de FDA utilisées avec les APDU de demande de SMK Device Annunciation.

Tableau 23 – Valeurs de l'adresse de FDA pour les APDU de demande de SMK device annunciation

Utilisation	Adresse de FDA LLLL.NN.SS	Adresse de réseau	Trajet de demande
Device Annunciation	00 00.00.02	SM Multicast	SMK de Type 5 -> Applications de Configuration

4.3.5.3.7.2 Paramètres d'APDU de demande

Le Tableau 24 définit les paramètres d'APDU de demande.

Tableau 24 – Paramètres d'APDU de demande de SMK device annunciation

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
SMK State	0	Unsigned8	1	Paramètre de Service Encodé comme suit. Bits 2-8: 0 = Reserved 1 = NO_TAG 2 = OPERATIONAL 3-127 = Reserved Bit 1: 0 = Not Synchronized with SNTP Time Server 1 = Synchronized with SNTP Time Server
Device Type	1	Unsigned8	1	Paramètre de Service Encodé comme suit. Chaque bit identifie un type différent d'appareil. C'est le type d'appareil de destination pour les messages de demande et l'appareil de source pour les messages de réponse. Une combinaison de bits 8-6 et 2,1 peut être établie pour les appareils. Le bit 5 n'est utilisé que pour les appareils de Type "X" et uniquement dans les réponses de SM Identity. Des non-appareils, qui participent dans l'attribution d'appareils, mettent à 0 la valeur 8 à 3 de ce paramètre. Bit 8 = Linking Device Bit 7 = I/O Gateway Bit 6 = Field Device Bit 5 = Type "X" Device Bit 4 = Reserved Bits 3-1 – Redundant Device Type Capability 0 = Non-redundant 1 = External Switchover Control 2 = External Switchover Control 3 = External and Internal Switchover Control 4 = Not used 5 = Internal Switchover Control and Non-redundant device 6 = External Switchover Control and Non-redundant device

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
				7 = External and Internal Switchover Control and Non-redundant device
Device Redundancy State	2	Unsigned8	2	Paramètre de Service Encodé comme suit. Encodé comme suit : Bit 8 = Réservé, mis à 0 Bit 7 = Réservé, mis à 0 Bit 6 = Réservé, mis à 0 Bit 5 = Réservé, mis à 0 Bits 4 et 3 – Device Redundancy Role 0 = Non-redundant 1 = Primary 2 = Secondary Bits 2 et 1 – Assigned Redundant Device Type 0 = Non-redundant 1 = External Switchover Control 2 = External Switchover Control
Duplicate Detection State	3	Unsigned8	2	Paramètre de Service Encodé comme suit. Bits 8-3: Réservé, mis à 0 Bit 2: 1 = PD Tag dupliqué détecté 0 = PD Tag dupliqué non détecté Bit 1: 1 = Device Device Index dupliqué détecté 0 = Device Index dupliqué non détecté
Device Index	4	Unsigned16	2	Paramètre de Service Encodé comme suit. 0 = Index non attribué 1 to Max Device Index = Device Index
Max Device Index	6	Unsigned16	2	Paramètre de Service
Operational Network Address	8	OctetString	16	Paramètre de Service
Device Id	24	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Physical Device Tag	56	VisibleString	32	Paramètre de Service Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Annunciation Repeat Time	92	Unsigned32	4	Paramètre de Service
LAN Redundancy Socket Address	96	Unsigned16	2	Paramètre de Service
Reserved	98	Unsigned16	2	Paramètre de Service
APDU Version Number	100	Unsigned32	4	Paramètre de Service
Device Version Number	100	Unsigned32	4	Paramètre de Service
Nombre d'entrées dans la liste des	104	Unsigned32	4	Ce paramètre indique combien de numéros de versions sont contenus dans le champ Version

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
numéros de version				Number List en dessous. Sa valeur est égale à 0, si ce n'est pas un appareil de liaison ou une passerelle.
Version Number List	108	Unsigned32	4 x Nombre d'entrées	Paramètre de Service Le champ n'est présent que pour des appareils de liaison et des passerelles I/O (Entrée/Sortie). Ce champ contient une liste de numéros. Chacun représente un numéro Unsigned32. Le nombre d'entrées dans la liste est spécifié par le champ au-dessus de "Number of Entries in Version Number List". Ce champ prend en charge deux types de listes tels que définis par la portion Link Id de FDA Address comme suit : Link Id = 0 Pour les appareils de liaison, il s'agit des numéros de la version d'une liste active de Type 9 dont l'un est prévu pour chaque interface de Type 9 attachée à l'appareil de liaison. Chaque élément de la liste est conçu comme suit : Bits 32-17: Type 9 Link Id Bits 16-9: Réservé, mis à 0 Bits 8-1: Version Number La liste est commandée par le numéro d'interface pontée de Type 9 avec Live List Version pour la première interface qui a lieu dans la liste. La valeur de 0 est utilisée pour les bits de Link Id si l'interface n'a pas été attribuée à une liaison Id. Link Id > 0 C'est une liste des numéros de version pour les adresses de nœud de Type "X", un numéro pour chaque nœud id dans la liste active pour la liaison de Type "X" spécifiée, y compris l'interface LD de Type "X. La liste des numéros de version des adresses de nœud de Type "X" n'est utilisée que lorsque ce format de message est utilisé pour des réponses de SM Identify qui décrivent une interface LD de Type "X". La liste commence par le numéro de la version spécifiée dans l'interface de Type "X". Le reste de la liste est classé par ordre croissant selon l'adresse de nœud de Type "X", avec deux numéros de version compressés dans chaque valeur Unsigned32 comme suit : Bits 33-25: Type “X” Node Address Bits 24-17 Version Number Bits 16-9: Type “X” Node Address Bits 8-1: Version Number

4.3.5.4 Services d’ASE de Type 9

4.3.5.4.1 Note générale

NOTE Sauf spécification contraire, tous les paramètres des APDU du Service de Type 9 et leurs valeurs sont définis dans les sous-séries de la CEI 61158-5.

4.3.5.4.2 Reject

Le service Reject de Type 9 n'est utilisé que pour indiquer au demandeur que la réponse à acheminer est trop longue pour le Type 9. L'APDU d'Erreur Typique achemine ces informations à l'aide de la classe d'erreur "Reject" et le code "Reject" tel que défini dans la

syntaxe abstraite pour les codes d'erreur au lieu de définir ici une APDU séparée pour acheminer ces informations.

4.3.5.4.3 Initiate (service confirmé Id = 96)

4.3.5.4.3.1 Paramètres d'APDU de demande

Le Tableau 25 définit les paramètres d'APDU de demande.

Tableau 25 – Paramètres d'APDU de demande initiale

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Connect Option	0	Unsigned8	1	Le champ indique l'option qui doit être utilisée pour ouvrir S-VFD Context. Ses valeurs sont : 1 = VCR Selector 2 = MIB Access 3 = FBAP Access
Access Protection Supported Calling	8	Boolean	1	Paramètre de Service
Password and Access Groups Calling	2	Unsigned16	2	Paramètre de Service
Version OD Calling	4	Integer16	2	Paramètre de Service
Profile Number Calling	6	Unsigned16	2	Paramètre de Service
PD Tag	8	OctetString	32	SMK Physical Device Tag de l'appareil contenant VFD de destination.

4.3.5.4.3.2 Paramètres d'APDU de réponse

Le Tableau 26 représente les paramètres d'APDU de réponse.

Tableau 26 – Paramètres d'APDU de réponse initiale

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Version OD Called	0	Integer16	2	Paramètre de Service
Profile Number Called	2	Unsigned16	2	Paramètre de Service

4.3.5.4.3.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1. Voir Codes d'erreur spécifiques initiaux de Type 9.

4.3.5.4.4 Abort (service non confirmé Id = 112)

4.3.5.4.4.1 Paramètres d'APDU de demande

Le Tableau 27 spécifie les paramètres d'APDU de demande.

Tableau 27 – Paramètres d'APDU de demande Abort

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Abort Detail	0	OctetString	16	Fourni par l'émetteur de l'APDU.
Abort Identifier	16	Unsigned8	1	Paramètre de Service. Valeurs définies pour les PDU de Type 9.
Reason Code	17	Unsigned8	1	Paramètre de Service
Reserved	18	Unsigned16	2	Réservé, mis à 0

4.3.5.4.5 Get status (service confirmé Id = 0)

4.3.5.4.5.1 Paramètres d'APDU de demande

Néant.

4.3.5.4.5.2 Paramètres d'APDU de réponse

Le Tableau 28 définit les paramètres d'APDU de réponse.

Tableau 28 – Paramètres d'APDU de réponse de Get

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Logical Status	0	Unsigned8	1	Paramètre de Service
Physical Status	1	Unsigned8	1	Paramètre de Service
Reserved	2	Unsigned16	2	Réservé, mis à 0
Local Detail	4	OctetString	4	Paramètre de Service. L'octet d'ordre supérieur est réservé et mis à 0. Les trois octets d'ordre inférieur contiennent la valeur Local Detail de 3 octets.

4.3.5.4.6 Status notification (service non confirmé Id = 1)

4.3.5.4.6.1 Paramètres d'APDU de demande

Les mêmes que pour les APDU de réponse de Get Status

4.3.5.4.7 Identify (service confirmé Id = 1)

4.3.5.4.7.1 Paramètres d'APDU de demande

Néant.

4.3.5.4.7.2 Paramètres d'APDU de réponse

Le Tableau 29 spécifie les paramètres d'APDU de réponse.

Tableau 29 – Paramètres d'APDU de réponse de Identify

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Nom de fournisseur	0	VisibleString	32	Paramètre de Service
Model Identifier	32	VisibleString	32	Paramètre de Service
Vendor Revision	64	VisibleString	32	Paramètre de Service

4.3.5.4.8 Get OD (service confirmé Id = 4)

4.3.5.4.8.1 Paramètres d'APDU de demande

Le Tableau 30 spécifie les paramètres d'APDU de demande.

Tableau 30 – Paramètres d'APDU de demande de Get OD

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Tout attribut	0	Boolean	1	Paramètre de Service
Start Index Flag	1	Boolean	1	Paramètre de Service
Reserved	2	Unsigned16	2	Réservé, mis à 0
Index	4	Unsigned32	4	Paramètre de Service

4.3.5.4.8.2 Paramètres d'APDU de réponse

Le Tableau 31 spécifie les paramètres d'APDU de réponse.

Tableau 31 – Paramètres d'APDU de réponse de Get OD

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
More Follows	0	Boolean	1	Paramètre de Service
Number of Object Descriptions	1	Unsigned8	1	Paramètre de Service
Reserved	2	Unsigned16	2	Réservé, mis à 0
Liste de Description d'Objets	4	OctetString	N	Défini en 4.3.5.6.2

4.3.5.4.8.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.9 Initiate Put OD (service confirmé Id = 28)

4.3.5.4.9.1 Paramètres d'APDU de demande

Le Tableau 32 spécifie les paramètres d'APDU de demande.

Tableau 32 – Paramètres d'APDU de demande de Initiate Put OD

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Reserved	0	Unsigned16	2	Réservé, mis à 0
Consequence	2	Unsigned16	2	Paramètre de Service. Valeurs définies pour les PDU de Type 9

4.3.5.4.9.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.9.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.10 Put OD (service confirmé Id = 29)

4.3.5.4.10.1 Paramètres d'APDU de demande

Le Tableau 33 spécifie les paramètres d'APDU de demande.

Tableau 33 – Paramètres d'APDU de demande de Put OD

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Number of Object Descriptions	0	Unsigned8	1	Nombre d'entrées dans List of Object Descriptions
Reserved	1	OctetString	3	Réservé, mis à 0
List of Object Descriptions	4	OctetString	N	Défini en 4.3.5.6.2

4.3.5.4.10.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.10.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.11 Terminate put OD (service confirmé Id = 30)

4.3.5.4.11.1 Paramètres d'APDU de demande

Néant.

4.3.5.4.11.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.11.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.

4.3.5.4.12 Generic initiate download sequence (service confirmé Id = 31)

4.3.5.4.12.1 Vue d'ensemble

Ce service correspond au Service Initiate Load de Type 9 avec les réglages des paramètres suivants :

- Index de la zone de chargement en appel: Zone de chargement du répondeur (Serveur)
- Type de chargement: TÉLÉCHARGEMENT DESCENDANT
- Service Load à utiliser: Push Segment (Generic Download Segment)
- Initiateur du Service Load: VRAI (Initiateur (Client) initie le service Load (le service Push Segment)).

4.3.5.4.12.2 Paramètres d'APDU de demande

Le Tableau 34 spécifie les paramètres d'APDU de demande.

Tableau 34 – Paramètres d'APDU de demande de Generic initiate download sequence

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.12.3 Paramètres d'APDU de réponse

Néant

4.3.5.4.12.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.13 Generic download segment (service confirmé Id = 32)

4.3.5.4.13.1 Vue d'ensemble

Ce service correspond au Service Generic Download Segment de Type 9.

4.3.5.4.13.2 Paramètres d'APDU de demande

Voir le Tableau 35.

Tableau 35 – Paramètres d'APDU de demande de Generic download segment

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
More Follows	4	Boolean	1	Paramètre de Service
Reserved	5	OctetString	3	Mis à 0.
Load Data	8	OctetString	N	Paramètre de Service

4.3.5.4.13.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.13.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.14 Generic terminate download sequence (service confirmé Id = 33)

4.3.5.4.14.1 Vue d'ensemble

Ce service correspond au Service Generic Terminate Download Sequence de Type 9.

4.3.5.4.14.2 Paramètres d'APDU de demande

Voir le Tableau 36.

Tableau 36 – Paramètres d'APDU de demande de Generic terminate download sequence

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.14.3 Paramètres d'APDU de réponse

Voir le Tableau 37.

Tableau 37 – Paramètres d'APDU de demande

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Terminate Reason	0	Boolean	1	Paramètre de Service
Reserved	1	OctetString	3	Réservé, mis à 0

4.3.5.4.14.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.15 Initiate download sequence (service confirmé Id = 9)

4.3.5.4.15.1 Vue d'ensemble

Ce service correspond au Service Initiate Download Sequence de Type 9 avec les réglages de paramètres suivants :

- Index de la zone de chargement en appel: Zone de chargement du répondeur (Serveur)
- Type de chargement: TÉLÉCHARGEMENT DESCENDANT
- Service Load à utiliser: Pull Segment (Download Segment)
- Initiateur du Service Load: FAUX (AP distant (Serveur) initie le service Load (le service Pull Segment)).

4.3.5.4.15.2 Paramètres d'APDU de demande

Voir le Tableau 38.

Tableau 38 – Paramètres d'APDU de demande de Initiate download sequence

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.15.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.15.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.16 Download segment (service confirmé Id = 10)

4.3.5.4.16.1 Vue d'ensemble

Ce service correspond au Service Download Segmentde Type 9.

4.3.5.4.16.2 Paramètres d'APDU de demande

Voir le Tableau 39.

Tableau 39 – Paramètres d'APDU de demande de Download segment

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.16.3 Paramètres d'APDU de réponse

Voir le Tableau 40.

Tableau 40 – Paramètres d'APDU de réponse de Download segment

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
More Follows	0	Boolean	1	Paramètre de Service
Reserved	2	OctetString	3	Réservé, chaque octet est mis à 0 binaire
Load Data	4	OctetString	N	Paramètre de Service

4.3.5.4.16.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.17 Terminate download sequence (service confirmé Id = 11)

4.3.5.4.17.1 Vue d'ensemble

Ce service correspond au Service Terminate Download Sequence de Type 9.

4.3.5.4.17.2 Paramètres d'APDU de demande

Voir le Tableau 41.

Tableau 41 – Paramètres d'APDU de demande de Terminate download sequence

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Reserved	4	OctetString	3	Réservé, chaque octet est mis à 0 binaire
Terminate Reason	7	Boolean	1	Paramètre de Service

4.3.5.4.17.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.17.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.18 Initiate upload sequence (service confirmé Id = 12)

4.3.5.4.18.1 Vue d'ensemble

Ce service correspond au Service Initiate Upload Sequence de Type 9 avec les réglages de paramètres suivants :

- Index de la zone de chargement en appel: Zone de chargement du répondeur (Serveur)

- Type de chargement: TÉLÉCHARGEMENT ASCENDANT
- Service Load à utiliser: Pull Segment (Upload Segment)
- Initiateur du Service Load: VRAI (Initiateur (Client) initie le service load (le service Pull Segment)).

4.3.5.4.18.2 Paramètres d'APDU de demande

Voir le Tableau 42.

Tableau 42 – Paramètres d'APDU de demande de Initiate upload sequence

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.18.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.18.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.19 Upload segment (service confirmé Id = 13)

4.3.5.4.19.1 Vue d'ensemble

Ce service correspond au Service Upload Sequence de Type 9.

4.3.5.4.19.2 Paramètres d'APDU de demande

Voir le Tableau 43.

Tableau 43 – Paramètres d'APDU de demande de Upload segment

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.19.3 Paramètres d'APDU de réponse

Voir le Tableau 44.

Tableau 44 – Paramètres d'APDU de réponse de Upload segment

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
More Follows	0	Boolean	1	Paramètre de Service
Reserved	2	OctetString	3	Réservé, chaque octet est mis à 0 binaire
Load Data	4	OctetString	N	Paramètre de Service

4.3.5.4.19.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.20 Terminate upload sequence (service confirmé Id = 14)

4.3.5.4.20.1 Vue d'ensemble

Ce service correspond au Service Terminate Upload Sequence de Type 9.

4.3.5.4.20.2 Paramètres d'APDU de demande

Voir le Tableau 45.

Tableau 45 – Paramètres d'APDU de demande de Terminate upload sequence

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.20.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.20.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.21 Request domain download (service confirmé Id = 15)

4.3.5.4.21.1 Vue d'ensemble

Ce service correspond au Service Request Domain Download de Type 9.

4.3.5.4.21.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 46.

Tableau 46 – Paramètres d'APDU de demande de Request domain download

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Additional Information	4	VisibleString	N	Paramètre de Service

4.3.5.4.21.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.21.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.22 Request domain upload (service confirmé Id = 16)

4.3.5.4.22.1 Vue d'ensemble

Ce service correspond au Service Create Program Invocation de Type 9.

4.3.5.4.22.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 47.

Tableau 47 – Paramètres d'APDU de demande de Request domain upload

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Additional Information	4	VisibleString	N	Paramètre de Service

4.3.5.4.22.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.22.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.23 Create program invocation (service confirmé Id = 17)

4.3.5.4.23.1 Vue d'ensemble

Create Program Invocation correspond au Service Create Program Invocation de Type 9.

4.3.5.4.23.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 48.

Tableau 48 – Paramètres d'APDU de demande de Create program invocation

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Reusable	0	Boolean	1	Paramètre de Service
Reserved	1	Unsigned8	1	Réservé, mis à 0
Number of Domain Indexes	2	Unsigned16	2	Le nombre d'index de domaine dans ce service
List Of Domain Indexes	4	Unsigned32	4 X Number of Domain Indexes	Paramètre de Service

4.3.5.4.23.3 Paramètres d'APDU de réponse

Voir les définitions dans le Tableau 49.

Tableau 49 – Paramètres d'APDU de réponse de Create program invocation

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.23.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.24 Delete program invocation (service confirmé Id = 18)

4.3.5.4.24.1 Vue d'ensemble

Delete Program Invocation correspond au Service Delete Program Invocation de Type 9.

4.3.5.4.24.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 50.

Tableau 50 – Paramètres d'APDU de demande de Delete program invocation

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.24.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.24.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.25 Start (service confirmé Id = 19)

4.3.5.4.25.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 51.

Tableau 51 – Paramètres d'APDU de demande de Start

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Execution Argument	4	OctetString	N	Paramètre de Service

4.3.5.4.25.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.25.3 Paramètres d'APDU de PI Error

Définis par les paramètres d'APDU de Common PI Error en 8.3.6.2.

4.3.5.4.26 Stop (service confirmé Id = 20)

4.3.5.4.26.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 52.

Tableau 52 – Paramètres d'APDU de demande de Stop

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.26.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.26.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU de Common PI Error en 8.3.6.2.

4.3.5.4.27 Resume (service confirmé Id = 21)

4.3.5.4.27.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 53.

Tableau 53 – Paramètres d'APDU de demande de Resume

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Execution Argument	4	OctetString	N	Paramètre de Service

4.3.5.4.27.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.27.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU de Common PI Error en 8.3.6.2.

4.3.5.4.28 Reset (service confirmé Id = 22)

4.3.5.4.28.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 54.

Tableau 54 – Paramètres d'APDU de demande de Reset

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.28.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.28.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU de Common PI Error en 8.3.6.2.

4.3.5.4.29 Kill (service confirmé Id = 23)

4.3.5.4.29.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 55.

Tableau 55 – Paramètres d'APDU de demande de Kill

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.29.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.29.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.30 Read (service confirmé Id = 2)

4.3.5.4.30.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 56.

Tableau 56 – Paramètres d'APDU de demande de Read

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.30.2 Paramètres d'APDU de réponse

Voir les définitions dans le Tableau 57.

Tableau 57 – Paramètres d'APDU de réponse de Read

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Valeur	0	OctetString	N	Paramètre de Service

4.3.5.4.30.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.31 Read with subindex (service confirmé Id = 82)

4.3.5.4.31.1 Vue d'ensemble

Ce service correspond au Service Read de Type 9.

4.3.5.4.31.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 58.

Tableau 58 – Paramètres d'APDU de demande de Read with subindex

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Component ID	4	Unsigned32	4	Paramètre de Service

4.3.5.4.31.3 Paramètres d'APDU de réponse

Voir les définitions dans le Tableau 59.

Tableau 59 – Paramètres d'APDU de réponse de Read with subindex

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Value	0	OctetString	N	Paramètre de Service

4.3.5.4.31.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.32 Write (service confirmé Id = 3)

4.3.5.4.32.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 60.

Tableau 60 – Paramètres d'APDU de demande de Write

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Value	4	OctetString	N	Paramètre de Service

4.3.5.4.32.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.32.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.33 Write with subindex (service confirmé Id = 83)

4.3.5.4.33.1 Vue d'ensemble

Ce service correspond au Service Write de Type 9.

4.3.5.4.33.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 61.

Tableau 61 – Paramètres d'APDU de demande de Write with subindex

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Component ID	4	Unsigned32	4	Paramètre de Service
Value	8	OctetString	N	Paramètre de Service

4.3.5.4.33.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.33.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.34 Define variable list (service confirmé Id = 7)

4.3.5.4.34.1 Vue d'ensemble

Ce service correspond au Service Define Variable List de Type 9.

4.3.5.4.34.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 62.

Tableau 62 – Paramètres d'APDU de demande de Define variable list

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Number of Indexes	0	Unsigned32	4	Nombre d'index variables dans la liste qui suit.
List Of Indexes	4	Unsigned32	4 x Number of Indexes	Paramètre de Service

4.3.5.4.34.3 Paramètres d'APDU de réponse

Voir les définitions dans le Tableau 63.

Tableau 63 – Paramètres d'APDU de réponse de Define variable list

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.34.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.35 Delete variable list (service confirmé Id = 8)

4.3.5.4.35.1 Vue d'ensemble

Ce service correspond au Service Delete Variable List de Type 9.

4.3.5.4.35.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 64.

Tableau 64 – Paramètres d'APDU de demande de Delete variable list

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service

4.3.5.4.35.3 Paramètres d'APDU de réponse

Néant.

4.3.5.4.35.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.36 Information report (service non confirmé = 0)

4.3.5.4.36.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 65.

Tableau 65 – Paramètres d'APDU de demande de Information report

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Value	4	OctetString	N	Paramètre de Service

4.3.5.4.37 Information report with subindex (service non confirmé = 16)

4.3.5.4.37.1 Vue d'ensemble

Ce service correspond au Service Information Report de Type 9.

4.3.5.4.37.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 66.

Tableau 66 – Paramètres d'APDU de demande de Information report

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Component ID	4	Unsigned32	4	Paramètre de Service
Value	8	OctetString	N	Paramètre de Service

4.3.5.4.38 Information report on change (service non confirmé Id = 17)

4.3.5.4.38.1 Vue d'ensemble

Ce service correspond au Service Information Report de Type 9.

Cette APDU est utilisée pour envoyer des données qui ont été éditées seulement après le changement de leur valeur. Le VCR qui contrôle l'édition indique si le service doit être utilisé.

4.3.5.4.38.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 67.

Tableau 67 – Paramètres d'APDU de demande de Information report on change

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Value	4	OctetString	N	Paramètre de Service

4.3.5.4.39 Information report on change with subindex (service non confirmé Id = 18)

4.3.5.4.39.1 Vue d'ensemble

Ce service correspond au Service Information Report de Type 9.

Cette APDU est utilisée pour envoyer des données qui ont été éditées seulement après le changement de leur valeur. Le VCR qui contrôle l'édition indique si le service doit être utilisé.

4.3.5.4.39.2 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 68.

Tableau 68 – Paramètres d'APDU de demande de Information report on change with subindex

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Component ID	4	Unsigned32	4	Paramètre de Service
Value	8	OctetString	N	Paramètre de Service

4.3.5.4.40 Event notification (service non confirmé Id = 2)

4.3.5.4.40.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 69.

Tableau 69 – Paramètres d'APDU de demande de Event notification

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Event Number	4	Unsigned32	4	Paramètre de Service
Data	8	OctetString	N	Paramètre de Service

4.3.5.4.41 Alter event condition monitoring (service confirmé Id = 24)

4.3.5.4.41.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 70.

Tableau 70 – Paramètres d'APDU de demande de Alter event condition monitoring

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Reserved	4	OctetString	3	Réservé, chaque octet est mis à 0 binaire
Enabled	7	Boolean	1	Paramètre de Service

4.3.5.4.41.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.41.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.4.42 Acknowledge event notification (service confirmé Id = 25)

4.3.5.4.42.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 71.

Tableau 71 – Paramètres d'APDU de demande de Acknowledge event notification

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Paramètre de Service
Event Number	4	Unsigned32	4	Paramètre de Service

4.3.5.4.42.2 Paramètres d'APDU de réponse

Néant.

4.3.5.4.42.3 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.5 Services de LAN redundancy

4.3.5.5.1 LAN redundancy diagnostic message (service non confirmé Id = 1)

4.3.5.5.1.1 Paramètres d'APDU de demande

Voir les définitions dans le Tableau 72.

Tableau 72 – Paramètres d'APDU de demande de LAN redundancy diagnostic message

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Device Index	0	Unsigned16	2	Paramètre de Service 0 = Index non attribué 1 – 65535 = Device Index
Number of Interfaces	2	Unsigned8	1	Paramètre de Service.
Transmission Interface	3	Unsigned8	1	Identifie l'interface à partir de laquelle cette APDU a été transmise 1 = Interface A 2 = Interface B
Diagnostic Message Interval	4	Unsigned 32	4	Paramètre de Service.
SMK Physical Device Tag	8	VisibleString	32	Paramètre de Service. Des octets qui ne sont pas utilisés dans ce champ sont établis sur les caractères espace de l'ASCII.
Reserved	40	Unsigned8	1	Reserved. Mis à 0
Duplicate Detection State	41	Unsigned8	1	Paramètre de Service. Encodé comme suit : Bits 8–3: Réservé, mis à 0 Bit 2: 1 = PD Tag dupliqué détecté 0 = PD Tag dupliqué non détecté Bit 1: 1 = Device Device Index dupliqué détecté 0 = Device Index dupliqué non détecté

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Number of Interface Statuses	42	Unsigned16	2	Le nombre d'entrées Unsigned32 dans les quatre Array of Interface Statuses qui suivent. La valeur de ce champ est 1/32 de la valeur de Max Device Index configurée à l'aide du service SM Set Assignment.
Array of Interface AtoA Statuses	44	Array of Unsigned32	4* Number of Interface Statuses	Chaque valeur de 32 bits représente le statut des 32 appareils. (voir la NOTE) 1 bit représente chaque appareil. Chaque bit représente le statut du trajet de transmission à partir de l'interface A de l'appareil d'alertes vers l'interface A de cet appareil. Les valeurs suivantes sont utilisées pour chaque bit : 0 = Messages de diagnostic réussis 1 = Messages de diagnostic ne sont pas reçus
Array of Interface AtoA Statuses	44 + 4* Number of Interface Statuses	Array of Unsigned32	4* Number of Interface Statuses	Chaque valeur de 32 bits représente le statut des 32 appareils. (voir la NOTE) 1 bit représente chaque appareil. Chaque bit représente le statut du trajet de transmission à partir de l'interface B de l'appareil d'alertes vers l'interface A de cet appareil. Les valeurs suivantes sont utilisées pour chaque bit : 0 = Messages de diagnostic réussis 1 = Messages de diagnostic ne sont pas reçus
Array of Interface AtoB Statuses	44 + 8* Number of Interface Statuses	Array of Unsigned32	4* Number of Interface Statuses	Chaque valeur de 32 bits représente le statut des 32 appareils. (voir la NOTE) 1 bit représente chaque appareil. Chaque bit représente le statut du trajet de transmission à partir de l'interface A de l'appareil d'alertes vers l'interface B de cet appareil. Les valeurs suivantes sont utilisées pour chaque bit : 0 = Messages de diagnostic réussis 1 = Messages de diagnostic ne sont pas reçus
Array of Interface BtoB Statuses	44 + 12* Number of Interface Statuses	Array of Unsigned32	4* Number of Interface Statuses	Chaque valeur de 32 bits représente le statut des 32 appareils. (voir la NOTE) 1 bit représente chaque appareil. Chaque bit indique le statut du trajet de transmission à partir de l'interface B de l'appareil d'alertes vers l'interface B de cet appareil. Les valeurs suivantes sont utilisées pour chaque bit : 0 = Messages de diagnostic réussis 1 = Messages de diagnostic ne sont pas reçus
NOTE Le bit qui identifie un appareil est localisé par sa position de bits dans le tableau. La division de Device Index de l'appareil par 32 génère l'index de l'élément de tableau Unsigned32 basé sur le zéro. Le reste de la division identifie le bit spécifique dans l'élément de tableau Unsigned32 à partir du bit de poids fort (= remainder 1). C'est pourquoi, un appareil avec Device Index 33 serait représenté par le bit de poids fort (bit 32) du deuxième élément de tableau Unsigned32, et un appareil avec Device Index 34 serait représenté par le bit de poids fort (bit 31) du deuxième élément de tableau Unsigned32.				

4.3.5.5.2 LAN redundancy get information (service confirmé Id = 1)

4.3.5.5.2.1 Vue d'ensemble

Toujours confirmé, ce service utilise Invoke Id dans la fin de trame de message.

4.3.5.5.2.2 Paramètres d'APDU de demande

Néant.

4.3.5.5.2.3 Paramètres d'APDU de réponse

Voir les définitions dans le Tableau 73.

Tableau 73 – Paramètres d'APDU de réponse de LAN redundancy get information

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
LAN Redundancy Attributes Version	0	Unsigned32	4	Paramètre de Service
Max Sequence Number Difference	4	Unsigned8	1	Paramètre de Service
LAN Redundancy Flags	5	Unsigned8	1	Paramètre de Service. Encodé comme suit ; 0 = Faux, 1 = Vrai Bit 1 Single Mcast APDU Tx I/F Validé Bit 2 Détection du câble croisé Validé Bit 3 Single Mcast APDU Rcpt I/F Validé Bit 4 Diagnostic à l'aide des propres APDU Validé Bit 5 Équilibrage de chargement Validé Bit 6-8 Réservé, mis à 0
Reserved	6	Unsigned16	2	Reserved. Mis à 0.
Diagnostic Message Interval	8	Unsigned32	4	Paramètre de Service
AgingTime	12	Unsigned32	4	Paramètre de Service
Diagnostic Message Interface A Send Address	16	OctetString	16	Ce paramètre fait partie des paramètres du service Diagnostic Message Send Addresses. Il définit l'adresse de destination pour les messages de diagnostic envoyés depuis l'interface A.
Diagnostic Message Interface A Receive Address	32	OctetString	16	Ce paramètre fait partie des paramètres du service Diagnostic Message Receive Addresses. Il définit l'adresse de destination pour les messages de diagnostic reçus depuis l'interface A.
Diagnostic Message Interface B Send Address	48	OctetString	16	Ce paramètre fait partie des paramètres du service Diagnostic Message Send Addresses. Il définit l'adresse de destination pour les messages de diagnostic envoyés depuis l'interface B.
Diagnostic Message Interface B Receive Address	64	OctetString	16	Ce paramètre fait partie des paramètres du service Diagnostic Message Receive Addresses. Il définit l'adresse de destination pour les messages de diagnostic reçus depuis l'interface B.

4.3.5.5.2.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.5.3 LAN redundancy put information (service confirmé Id = 2)

4.3.5.5.3.1 Vue d'ensemble

Toujours confirmé, ce service utilise Invoke Id dans la fin de trame de message.

4.3.5.5.3.2 Paramètres d'APDU de demande

Les mêmes que les paramètres de réponse de LAN Redundancy Get Information. Seuls les champs suivant le numéro de la version d'APDU sont utilisés par l'entité réceptrice de LAN Redundancy.

4.3.5.5.3.3 Paramètres d'APDU de réponse

Les mêmes que les paramètres de Request Message. La version d'attributs de LAN Redundancy est incluse.

4.3.5.5.3.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.5.4 LAN redundancy get statistics (service confirmé Id = 3)

4.3.5.5.4.1 Vue d'ensemble

Toujours confirmé, ce service utilise Invoke Id dans la fin de trame de message.

4.3.5.5.4.2 Paramètres d'APDU de demande

Néant.

4.3.5.5.4.3 Paramètres d'APDU de réponse

Voir les définitions dans Tableau 74.

Tableau 74 – Paramètres d'APDU de demande de LAN redundancy get statistics

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Diagnostic Messages Received On Interface A	0	Unsigned32	4	Paramètre de Service
Diagnostic Messages Missed On Interface A	4	Unsigned32	4	Paramètre de Service
Remote Device Diagnostic Message Receive Faults Detected On Interface A	8	Unsigned32	4	Paramètre de Service
Diagnostic Messages Received On Interface B	12	Unsigned32	4	Paramètre de Service
Diagnostic Messages Missed On Interface B	16	Unsigned32	4	Paramètre de Service
Remote Device Diagnostic Message Receive Faults Detected On Interface B	20	Unsigned32	4	Paramètre de Service
Number of Crossed Cable	24	Unsigned32	4	Ce champ représente le numéro des valeurs Unsigned32 dans la liste Crossed Cable Status. 0

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Statuses				signifie que la liste n'est pas présente. Si la liste est présente, le champ contient 1/32 de la valeur de Max Device Index attribuée à l'appareil dans le message de SM Set Assignment Info.
Array of Crossed Cable Status	28	Array of Unsigned32	4 * Number of Crossed Cable Statuses	Paramètre de Service. Chaque bit dans ce tableau (voir la NOTE) représente un appareil qui envoie des messages de diagnostic à l'appareil qui est en cours d'envoi de ce message (appareil d'alertes). Chaque bit de l'appareil distant indique si l'interface qu'il utilise pour envoyer des messages de diagnostic (tel qu'indiqué par l'interface utilisée pour la transmission de ce champ de messages de diagnostic) est la même interface utilisée par l'appareil d'alertes pour recevoir les messages. Les valeurs suivantes sont utilisées pour chaque bit : 0 = Les câbles ne sont pas croisés ou la détection validée de câbles croisés est fausse. 1 = Les câbles sont croisés (Message de diagnostic envoyé depuis l'Interface A a été reçu sur l'Interface B, soit un message de diagnostic envoyé de l'Interface B a été reçu sur l'Interface A).
NOTE Le bit identifiant un appareil est localisé par sa position de bits dans le tableau. La division de Device Index de l'appareil par 32 génère l'index de l'élément de tableau Unsigned32 basé sur le zéro. Le reste de la division identifie le bit spécifique dans l'élément de tableau Unsigned32 à partir du bit de poids fort (= remainder 1). C'est pourquoi, un appareil avec Device Index 33 serait représenté par le bit de poids fort (bit 32) du deuxième élément de tableau Unsigned32, et un appareil avec Device Index 34 serait représenté par le bit de poids fort (bit 31) du deuxième élément de tableau Unsigned32.				

4.3.5.5.4.4 Paramètres d'APDU d'erreur

Définis par les paramètres d'APDU d'Erreur Typique en 8.3.6.1.

4.3.5.6 Structure des descriptions d'objets

4.3.5.6.1 Généralités

Le Paragraphe 4.3.5.6 définit les descriptions d'objets acheminées à l'aide des services Get OD et Put OD de Type 5. Les VFD employant les index OD de 32 bits sont indispensables pour utiliser les formats HSE de description d'objets définis ci-dessous. Les VFD employant les index OD de 16 bits peuvent utiliser soit le format H1, soit le format HSE de description d'objet.

L'Agent FDA peut convertir les descriptions d'objet formatées de Type 9 qu'il reçoit depuis les VFD ou les appareils de Type 9 en formats de Type 5 correspondants, mais cela n'est pas obligatoire. D'ailleurs, sans que ce soit obligatoire, l'Agent de FDA peut convertir les descriptions d'objets formatées HSE, qu'il reçoit à partir des messages de demande de PutOD de Type 5, en format de Type 9 pour la livraison ou le suivi vers les VFD qui utilisent les formats de Type 9.

Lorsqu'un Agent de FDA, qui ne prend pas en charge la conversion des formats de Type 5 en Type 9, reçoit un tel message de demande nécessitant une conversion, il peut facilement faire suivre le message et compter sur le VFD récepteur pour retourner le message d'erreur approprié. Il peut également retourner une réponse négative avec une classe d'erreur "od" et un code d'erreur ="type 5 to type 9 format conversion not supported".

Les champs contenus dans chaque concept de OD ci-dessous sont définis dans la Spécification de Type 9.

Leurs définitions sont les mêmes ici, pourtant, leurs champs *index, data type* et *longueur* sont passés à 32 bits, et leur ordre a changé pour s'aligner sur les limites de 32 bits.

Pour les champs définis avec les valeurs Boolean, la valeur 0 indique "FAUX" et la valeur non nulle correspond à "VRAI".

Pour maintenir la compatibilité avec les descriptions d'objets de Type 9, le premier champ dans une description d'objets acheminée par l'Agent de FDA est OD Tag. OD Tag contient soit l'indicateur d'une description d'objets de Type 9, soit un indicateur réservé pour les descriptions d'objets de Type 5 (valeur hexadécimale 0xFFFF). Le suivi de OD Tag représente la description d'objets indiquée par l'indicateur. Chaque description d'objets de Type 5 utilise l'un des formats spécifiés en 4.3.5.6.

Chaque description d'objets de Type 9 contient un indicateur de Type 5 et une description d'objet de Type 5 telle que spécifiée par le Type 9. L'indicateur de Type 5 est l'indicateur ASN.1 de Type 9 encodé et spécifié pour les messages de réponse de GetOD et de demande de PutOD. La description d'objets de Type 9 est spécifiée sous forme des données compressées.

L'indicateur de Type 9 est composé d'un ou de deux octets. La valeur des premiers quatre bits de poids fort du premier octet (octet de poids fort) est égale à 0. Les quatre bits suivants correspondent à la longueur de la description d'objets. Si ce champ de 4 bits de longueur contient une valeur inférieure à la valeur hexadécimale 0xF, l'indicateur H1 correspond, dans ce cas, à un octet et le champ de 4 bits de longueur spécifie la longueur de la description d'objets qui suit.

Après le champ "Type Flag "de Type 5, toutes les descriptions d'objets de Type 5 possèdent cinq champs supplémentaires. Le format de cet en-tête typique pour la description d'objets de Type 5 est représenté dans le Tableau 75.

Tableau 75 – En-tête de la description d'objets

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets	Description
OD Tag	0	Unsigned16	2	Établi à $FFFF_{16}$ pour indiquer une Description d'Objet de Type 5
Object Code	2	Integer8	1	Code d'objet pour un Objet
All Attributes Flag	3	Unsigned8	1	Indique une forme longue (=1) ou courte (=0) Description d'Objet
Object Description Length	4	Unsigned16	2	Longueur totale en octets de la Description d'Objet
Reserved (mis à zéro)	6	Unsigned16	2	Mis à 0
Index32	8	Unsigned32	4	Index d'Objet de 32 bits

Un objet non valide pour le Type 5 est utilisé pour indiquer un index d'objet inutilisé comme c'est le cas pour le Type 9. Il est représenté dans le Type 5 tel que montré dans le Tableau 76.

Tableau 76 – Objet non valide

Nom du champ	Valeur
OD Tag	$FF\ FF_{16}$
Object Code	0
All Attributes Flag	0
Object Description Length	12
Reserved (mis à zéro)	0
Index32	Index of Null Object

4.3.5.6.2 Structure de la liste des descriptions d'objets

Voir les définitions dans le Tableau 77.

Tableau 77 – Structure de la liste des descriptions d'objets

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 16	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
ROM/RAM Flag	12	Unsigned8	1
Access Protection Supported	13	Unsigned8	1
Reserved (mis à zéro)	14	Unsigned16	2
Name Length	16	Unsigned32	4
Version OD	20	Unsigned32	4
ST-OD Length	24	Unsigned32	4
First Index S-OD	28	Unsigned32	4
S-OD Length	32	Unsigned32	4
First Index DV-OD	36	Unsigned32	4
DV-OD Length	40	Unsigned32	4
First Index DP-OD	44	Unsigned32	4
DP-OD Length	48	Unsigned32	4
Champs supplémentaires de forme longue (voir la NOTE)	52		
Local Address OD-ODES	0 + Long Form Offset	Unsigned64	8
Local Address ST-OD	8 + Long Form Offset	Unsigned64	8
Local Address S-OD	16 + Long Form Offset	Unsigned64	8
Local Address DV-OD	24 + Long Form Offset	Unsigned64	8
Local Address DP-OD	32 + Long Form Offset	Unsigned64	8
NOTE Ces champs ne sont pas présents, si la forme courte est demandée.			

4.3.5.6.3 Structure d'une zone de chargement dans le S-OD

Voir les définitions dans le Tableau 78.

Tableau 78 – Structure d'une zone de chargement dans le S-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 18	2	Integer8	1
All Attributes Flag	3	Unsigned8	1

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Domain State	12	Unsigned8	1
Upload State	13	Unsigned8	1
Reserved (mis à zéro)	14	Unsigned16	2
Max Octets	16	Unsigned32	4
Counter	20	Unsigned32	4
Champs supplémentaires de forme longue (voir la NOTE 1)	24		
Local Address	0 + Long Form Offset	Unsigned64	8
Password	8 + Long Form Offset	Unsigned8	1
Access Groups	9 + Long Form Offset	Unsigned8	1
Access Rights	10 + Long Form Offset	Unsigned16	2
Extension Length	12 + Long Form Offset	Unsigned32	4
Extension Data	16 + Long Form Offset	OctetString	Extension Length
Domain Name	16 + Long Form Offset + Extension Length	VisibleString	Name Length (voir la NOTE 2)
NOTE 1 Ces champs ne sont pas présents, si la forme courte est demandée.			
NOTE 2 Ce champ n'est pas présent, si Name Length dans OD Object Description est égal à 0.			

4.3.5.6.4 Structure d'une invocation de fonction dans le DP-OD

Voir les définitions dans le Tableau 79.

Tableau 79 – Structure d'une invocation de fonction dans le DP-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 19	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Deletable	12	Unsigned8	1
Reusable	13	Unsigned8	1
PI-State	14	Unsigned8	1
Reserved	15	Unsigned8	3
Number Of Domains	16	Unsigned32	4
List of Domain Indexes	20	Unsigned32	Number Of Domains x 4 (voir la NOTE 1)
Champs supplémentaires de forme longue (voir la NOTE 2)	20 + (Number Of Domains x 4)		
Password	0 + Long Form Offset	Unsigned8	1

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Access Groups	1 + Long Form Offset	Unsigned8	1
Access Rights	2 + Long Form Offset	Unsigned16	2
Extension Length	4 + Long Form Offset	Unsigned32	4
Extension Data	8 + Long Form Offset	OctetString	Extension Length
PI Name	8 + Long Form Offset + Extension Length	VisibleString	Name Length (voir la NOTE 3)
NOTE 1 Ce champ n'est pas présent, si Number of Domains est égal à 0.			
NOTE 2 Ces champs ne sont pas présents, si la forme courte est demandée.			
NOTE 3 Ce champ n'est pas présent, si Name Length dans OD Object Description est égal à 0.			

4.3.5.6.5 Structure d'un événement dans le S-OD

Voir les définitions dans le Tableau 80.

Tableau 80 – Structure d'un événement dans le S-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 20	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Reserved	12	OctetString	3
Enabled	15	Unsigned8	1
Index Event Data	16	Unsigned32	4
Length	20	Unsigned32	4
Champs supplémentaires de forme longue (voir la NOTE 1)	24		
Password	0 + Long Form Offset	Unsigned8	1
Access Groups	1 + Long Form Offset	Unsigned8	1
Access Rights	2 + Long Form Offset	Unsigned16	2
Extension Length	4 + Long Form Offset	Unsigned32	4
Extension Data	8 + Long Form Offset	OctetString	Extension Length
Event Name	8 + Long Form Offset + Extension Length	VisibleString	Name Length (voir la NOTE 2)
NOTE 1 Ces champs ne sont pas présents, si la forme courte est demandée.			
NOTE 2 Ce champ n'est pas présent, si Name Length dans OD Object Description est égal à 0.			

4.3.5.6.6 Structure d'un type de données dans le ST-OD

Voir les définitions dans le Tableau 81.

Tableau 81 – Structure d'un type de données dans le ST-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 21	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Reserved	10	Unsigned16	2
Length of Data Type	12	Unsigned32	4
Symbolic Name Length	16	Unsigned32	4
*Symbolic Name	20	VisibleString	Symbolic Name Length
NOTE 1 Ce champ n'est pas présent, si Symbolic Name Length est égal à 0.			

4.3.5.6.7 Structure de la description d'un type de données dans le ST-OD

Voir les définitions dans le Tableau 82.

Tableau 82 – Structure d'une description d'un type de données dans le ST-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 22	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Reserved (mis à zéro)	12	Unsigned16	2
Number Of Elements	14	Unsigned16	2
List of Elements	16	Structure	8 x (Number of Elements)
Data Type Index		Unsigned32	4
Length		Unsigned32	4

4.3.5.6.8 Structure d'une simple variable dans le S-OD

Voir les définitions dans le Tableau 83.

Tableau 83 – Structure d'une simple variable dans le S-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 23	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Data Type Index	12	Unsigned32	4
Length	16	Unsigned32	4
Champs supplémentaires de forme longue (voir la NOTE 1)	20		
Local Address	0 + Long Form Offset	Unsigned64	8
Password	8 + Long Form Offset	Unsigned8	1
Access Groups	9 + Long Form Offset	Unsigned8	1
Access Rights	10 + Long Form Offset	Unsigned16	2
Extension Length	12 + Long Form Offset	Unsigned32	4
Extension Data	16 + Long Form Offset	OctetString	Extension Length
Variable Name	16 + Long Form Offset + Extension Length	VisibleString	Name Length (voir la NOTE 2)
NOTE 1 Ces champs ne sont pas présents, si la forme courte est demandée.			
NOTE 2 Ce champ n'est pas présent, si Name Length dans OD Object Description est égal à 0.			

4.3.5.6.9 Structure d'un tableau dans le S-OD

Voir les définitions dans le Tableau 84.

Tableau 84 – Structure d'un tableau dans le S-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 24	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Number Of Elements	12	Unsigned32	2
Data Type Index	16	Unsigned32	4
Length	20	Unsigned32	4
Champs supplémentaires de forme longue (voir la NOTE 1)	24		
Local Address	0 + Long Form Offset	Unsigned64	8

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Password	8 + Long Form Offset	Unsigned8	1
Access Groups	9 + Long Form Offset	Unsigned8	1
Access Rights	10 + Long Form Offset	Unsigned16	2
Extension Length	12 + Long Form Offset	Unsigned32	4
Extension Data	16 + Long Form Offset	OctetString	Extension Length
Variable Name	16 + Long Form Offset + Extension Length	VisibleString	Name Length (voir la NOTE 2)
NOTE 1 Ces champs ne sont pas présents, si la forme courte est demandée.			
NOTE 2 Ce champ n'est pas présent, si Name Length dans OD Object Description est égal à 0.			

4.3.5.6.10 Structure d'un enregistrement dans le S-OD

Voir les définitions dans le Tableau 85.

Tableau 85 – Structure d'un enregistrement dans le S-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 25	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Data Type Index	12	Unsigned32	4
Reserved	16	Unsigned32	4
Champs supplémentaires de forme longue (voir la NOTE 1)	20		
List of Local Addresses	0 + Long Form Offset	Unsigned64	K x 8 (voir la NOTE 2)
Password	(K x 8) + Long Form Offset	Unsigned8	1
Access Groups	1 + (K x 8) + Long Form Offset	Unsigned8	1
Access Rights	2 + (K x 8) + Long Form Offset	Unsigned16	2
Extension Length	4 + (K x 8) + Long Form Offset	Unsigned32	4
Extension Data	8 + (K x 8) + Long Form Offset	OctetString	Extension Length
Record Name	8 + (K x 8) + Long Form Offset + Extension Length	VisibleString	Name Length (voir la NOTE 3)
NOTE 1 Ces champs ne sont pas présents, si la forme courte est demandée.			
NOTE 2 K représente le nombre de champs dans l'enregistrement.			
NOTE 3 Ce champ n'est pas présent, si Name Length dans OD Object Description est égal à 0.			

4.3.5.6.11 Structure d'une liste de variables dans le DV-OD

Voir les définitions dans le Tableau 86.

Tableau 86 – Structure d'une liste de variables dans le DV-OD

Nom du champ	Décalage d'octets	Type de données	Longueur d'octets
Champs de forme courte			
OD Tag = FF FF_{16}	0	Unsigned16	2
Object Code = 26	2	Integer8	1
All Attributes Flag	3	Unsigned8	1
Object Description Length	4	Unsigned16	2
Reserved (mis à zéro)	6	Unsigned16	2
Index32	8	Unsigned32	4
Deletable	12	Unsigned8	1
Reserved	13	Unsigned8	1
Number Of Elements	14	Unsigned16	2
List of Element Indexes	16	Unsigned32	4 x Number of Elements
Champs supplémentaires de forme longue (voir la NOTE 1)	20 + (K x 4) (voir la NOTE 2)		
Password	0 + Long Form Offset	Unsigned8	1
Access Groups	1 + Long Form Offset	Unsigned8	1
Access Rights	2 + Long Form Offset	Unsigned16	2
Extension Length	4 + Long Form Offset	Unsigned32	4
Extension Data	8 + Long Form Offset	OctetString	Extension Length
Variable List Name	8 + Long Form Offset + Extension Length	VisibleString	Name Length (voir la NOTE 3)
NOTE 1 Ces champs ne sont pas présents, si la forme courte est demandée.			
NOTE 2 K représente Number of Elements dans l'enregistrement.			
NOTE 3 Ce champ n'est pas présent, si Name Length dans OD Object Description est égal à 0.			

4.3.6 Corps des APDU d'erreur

4.3.6.1 Paramètres d'erreurs typiques

Le format d'erreur suivant tel que représenté dans le Tableau 87 est utilisé pour de nombreux services.

Tableau 87 – Paramètres d'erreurs typiques

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Error Class	0	Unsigned8	1	Ce paramètre identifie la catégorie de l'erreur. Ses valeurs sont définies dans la production de Error Class de la Syntaxe Abstraite de Type 9. Les valeurs sont des valeurs marquées. La valeur 9 a été ajoutée pour l'utilisation des APDU de Initiate Error de Type 5.
Error Code	1	Unsigned8	1	Ce paramètre identifie le type d'erreur spécifique au sein de la classe. Ses valeurs sont définies dans la production de Error Class de la Syntaxe Abstraite de Type 9. Les valeurs sont les valeurs énumérées avec des valeurs entre parenthèses, telles que autre = 0. Des valeurs pour une classe d'erreur 11 ont été ajoutées pour l'utilisation de messages Initiate Error.
Additional Code	3	Integer16	1	Ce paramètre fournit un code supplémentaire associé à une erreur.
Additional Description	4	VisibleString	16	Ce paramètre fournit 16 octets d'informations supplémentaires associées à une erreur.

4.3.6.2 Paramètres d'erreur PI

Voir les définitions dans le Tableau 88.

Tableau 88 – Paramètres d'erreur PI

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Pi State	0	Unsigned8	1	Défini dans Paramètre de Service.
Reserved	1	OctetString	3	Réservé, établi à 0
Error Class	4	Unsigned8	1	Défini dans Paramètre de Service.
Error Code	5	Unsigned8	1	Défini dans Paramètre de Service.
Additional Code	7	Integer16	1	Défini dans Paramètre de Service.
Additional Description	8	VisibleString	16	Défini dans Paramètre de Service.

4.3.6.3 Paramètres d'erreur OD

Voir les définitions dans le Tableau 89.

Tableau 89 – Paramètres d'erreur OD

Nom du paramètre	Décalage d'octets	Type de données	Longueur d'octets	Description
Index	0	Unsigned32	4	Défini dans Paramètre de Service.
Error Class	4	Unsigned8	1	Défini dans Paramètre de Service.
Error Code	5	Unsigned8	1	Défini dans Paramètre de Service.
Additional Code	7	Integer16	1	Défini dans Paramètre de Service.
Additional Description	8	VisibleString	16	Défini dans Paramètre de Service.

4.3.6.4 Valeurs du code d'erreur et de la classe d'erreur

Le Tableau 90 spécifie les codes d'erreur de l'Agent de FDA pour chaque classe d'erreur.

Tableau 90 – Valeurs du code d'erreur et de la classe d'erreur

Description :		Valeurs :	
Classe	Code	Classe	Code
vfd state	autre	1	0
application reference	autre	2	0
	application inaccessible	2	1
definition	autre	3	0
	objet non défini	3	1
	attributs d'objet incohérents	3	2
	nom existe déjà	3	3
resource	autre	4	0
	mémoire indisponible	4	1
	demandes max par session dépassées	4	2
	sessions max dépassées	4	3
	échec de création d'objet	4	4
service	autre	5	0
	conflit d'états de l'objet	5	1
	taille de pdu	5	2
	conflit de contrainte de l'objet	5	3
	paramètre incohérent	5	4
	paramètre illégal	5	5
	service non pris en charge	5	6
	version non prise en charge	5	7
	options non valides	5	8
	protocole non pris en charge	5	9
	réservé	5	10
	incohérence des paramètres clé	5	11
	attributions déjà effectuées	5	12
	état de redondance de l'appareil non pris en charge	5	13
	délai d'attente de réponse	5	14

Description :		Valeurs :	
Classe	Code	Classe	Code
	PD Tag dupliqué détecté	5	15
access	autre	6	0
	objet non validé	6	1
	défaut de matériel	6	2
	accès à l'objet refusé	6	3
	adresse non valide	6	4
	attribut d'objet incohérent	6	5
	accès à l'objet non pris en charge	6	6
	objet inexistant	6	7
	conflit type	6	8
	accès nommé non pris en charge	6	9
	accès à l'élément non pris en charge	6	10
	accès à la configuration déjà ouvert	6	11
	réservé	6	12
	adresse de FDA non reconnue	6	13
Od	autre	7	0
	dépassement de la longueur du nom	7	1
	dépassement de od	7	2
	lecture od protégée	7	3
	dépassement de la longueur d'extension	7	4
	dépassement de la longueur de description od	7	5
	problème opérationnel	7	6
	conversion du format de type 5 en format de type 9 non prise en charge	7	7
Other	autre	8	0
Reject	taille de pdu	9	5
sm reason code	autre	10	0
	DLL Error – ressources insuffisantes	10	1
	DLL Error – queue d'envoi remplie	10	2
	DLL Error – délai d'attente avant transmission	10	3
	DLL Error – raison non spécifiée	10	4
	Appareil n'a pas réussi à répondre à SET_PD_TAG	10	5
	Appareil n'a pas réussi à répondre à WHO_HAS_PD_TAG	10	6
	Appareil n'a pas réussi à répondre à SET_ADDR	10	7
	Appareil n'a pas réussi à répondre à IDENTIFY	10	8
	Appareil n'a pas réussi à répondre à ENABLE_SM_OPU	10	9
	Appareil n'a pas réussi à répondre à CLEAR_ADDRESS	10	10
	Réponse multiple de WHO_HAS_PD_TAG	10	11
	Non-Matching PD_TAG from WHO_HAS_PD_TAG	10	12
	Non-Matching PD_TAG from IDENTIFY	10	13
	Non-Matching DEV_ID from IDENTIFY	10	14

Description :		Valeurs :	
Classe	Code	Classe	Code
	Remote Error État non valide	10	15
	Remote Error PD-Tag ne correspond pas	10	16
	Remote Error Dev-ID ne correspond pas	10	17
	Remote Error Échec d'écriture d'objet SMIB	10	18
	Remote Error Starting SM Operational	10	19
initiate	autre	11	0
	max-fms-pdu-size-insufficient	11	1
	feature-not-supported	11	2
	version-od-incompatible	11	3
	user-initiate-denied	11	4
	password-error	11	5
	profile-number-incompatible	11	6

4.4 Structure du diagramme d'états du protocole de FAL

Application de la structure du diagramme d'états du protocole de FAL pour le Type 9.

4.5 Diagramme d'états de SMK

4.5.1 Généralités

La machine protocolaire de SMK (SMKPM) définit le comportement du SMK pour les états qui affectent ses communications.

Dans la machine protocolaire qui suit :

1. Le paramètre Invoke Id n'est pas inclus de manière explicite. En revanche, il est considéré comme une partie intégrale de tous les services confirmés et des primitives Find Tag. Il est reçu de la part de l'utilisateur de SMK expéditeur et est acheminé implicitement à l'utilisateur de SMK distant.
2. L'adresse de réponse n'est pas incluse de manière explicite. Elle est toujours obtenue à partir de l'adresse du réseau de source de la demande.
3. Des AR ne sont pas utilisées pour envoyer et recevoir des APDU de SMK. En revanche, le SMKPM effectue les fonctions d'AR nécessaires.
4. La double barre oblique, c'est-à-dire "//", indique le début d'un commentaire dans une ligne.

4.5.2 Messages reçus par la SMKPM

La SMKPM reçoit des APDU issues de l'Agent de FDA. Ces APDU sont modélisées sous forme d'Événements des transitions des SMKPM. La fonction RcvMsg() est utilisée pour représenter l'arrivée d'une APDU.

4.5.3 Les primitives du service de SMKPM, les AR locales, les processeurs du service de SMK de Type 5, et les fonctions d'interface de Type "X"

Le Tableau 91 représente les primitives de services (par ordre alphabétique) utilisées dans la SMKPM. Elles sont utilisées comme suit :

- Les primitives de services de demande sont utilisées comme des Événements des transitions de SMKPM. Elles sont reçues à partir des AP locales.

- Les primitives de services de demande sont également utilisées dans les Actions des transitions de SMKPM. Quand elles sont utilisées de cette manière, elles génèrent la livraison d'un événement correspondant vers la SMKPM et sont traitées immédiatement en passant devant tous les autres événements en file d'attente.
- Les primitives de services d'indication sont utilisées dans les Actions des transitions de SMKPM. Elles peuvent être livrées à des AP locales ou à des Fonctions d'Interface de Type "X". Il existe une fonction d'interface pour chaque interface de Type "X" dans un appareil de liaison. Chacune d'entre elles est identifiée par la liaison id de FDA Address contenue dans une APDU reçue. Link Id 00 00 utilisé avec l'adresse de groupe NN.SS SMK de Type "X" signifie toutes les Fonctions d'Interface de Type "X". Les noms des services précédés par "LD_" identifient des services fournis par les Fonctions d'Interface de Type "X". Ces services obtiennent le paramètre sm_svc de l'APDU reçue. Ils invoquent un ou plusieurs services de SM de Type "X" pour satisfaire aux demandes reçues à partir du SMK de Type 5. L'émission d'une de ces demandes de service génère la sauvegarde de Invoke Id par le SMK de Type 5 pour pouvoir faire correspondre la réponse à la demande. Les services de Type "X" utilisés et leur séquence dépendent de la mise en œuvre. Si la Fonction d'Interface de Type "X" reçoit une demande LD_FindTagQuery et localise l'objet interrogé, elle répond en émettant une demande de SM_FindTagReply.
- Les primitives de services de réponse sont utilisées comme des Événements des transitions de SMKPM. Elles peuvent être reçues à partir d'une AR locale ou des Fonctions d'Interface.
- Les primitives de services de réponse sont également utilisées dans les Actions des transitions de SMKPM. Quand elles sont utilisées de cette manière, elles génèrent la livraison d'un événement correspondant vers la SMKPM et sont traitées immédiatement en passant devant tous les autres événements en file d'attente.
- Les primitives de services de confirmation sont utilisées dans les Actions des transitions de SMKPM. Elles sont livrées à des AP demandeurs locaux.

Tableau 91 – Primitives du service de SMKPM

Service	Primitives			
	.req	.ind	.rsp[a]	.cnf
Type_X_ClearAddress		X	X	
Type_X_ClearAssignmentInfo		X	X	
Type_X_FindTagQuery		X		
Type_X_FindTagReply		X		
Type_X_SetAssignmentInfo		X	X	
SM_ClearAddress	X	X	X	X
SM_ClearAssignmentInfo	X	X	X	X
SM_DeviceAnnunciation	X			
SM_FindTagQuery	X	X		
SM_FindTagReply	X	X		
SM_Identify	X	X	X	X
SM_SetAssignmentInfo	X	X	X	X

[a]. rsp est la primitive de réponse positive aussi bien que la primitive de réponse négative.

4.5.4 Traitement des transitions de tables d'états

Les tables d'états suivantes sont composées d'une série de transitions numérotées. L'ordre du traitement des transitions est basé sur les règles suivantes.

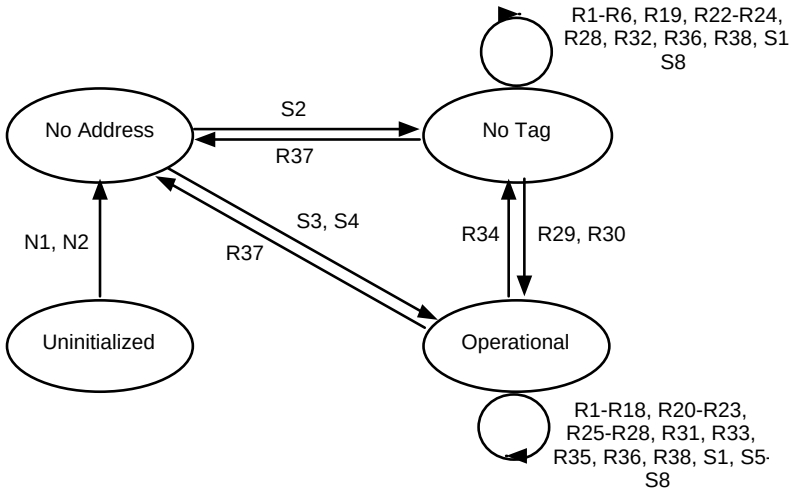
- Au moment de recevoir un événement, localiser la première transition dans la table contenant une instance de l'événement. Un événement de communications est identifié par son indication, c'est-à-dire SM_ClearAdress.ind.
- En commençant par cette transition, continuer vers le bas de la table d'états, transition par transition, jusqu'à ce qu'une transition soit localisée où l'État courant correspond à l'État courant du diagramme d'états et son Événement correspond à l'événement reçu.
- Tester les conditions. Si le résultat est faux, continuer vers le bas de la table d'états jusqu'à la transition suivante qui correspond à l'État courant et à l'Événement.
- Si aucune transition ne correspond, ignorer l'événement.
- Une fois la localisation effectuée, passer à l'action spécifique et changer l'État courant du diagramme d'états par l'état Suivant spécifié par la transition. Ensuite, attendre l'événement suivant.

4.5.5 États

Le Tableau 92 et la Figure 1 définissent les états et leurs transitions.

Tableau 92 – États de SMKPM

No Address	L'appareil n'a pas d'adresse de réseau sur ses interfaces de Type 5. Il n'est pas dans cet état, s'il a une adresse de réseau sur ses interfaces de Type 5. Dans cet état, la SMKPM attend un événement qui reçoit une adresse de réseau via une interface locale à partir des couches sous-jacentes.
No Tag	L'appareil n'a pas d'informations d'attribution valides. Dans cet état, le SMK envoie des demandes d'annonciation et écoute des APDU valides de Set Assignment Info. Il répond également à Clear Address et à des APDU de demande de SM Identify.
Operational	L'appareil a une adresse de réseau et des informations d'attribution valides. C'est l'état de fonctionnement normal du SMK.



Légende

Anglais	Français
No Adress	Pas d'adresse
No Tag	Pas de marqueur
Uninitialized	Non initialisé
Operational	Opérationnel

Figure 1 – Diagramme de transition d'états pour SMK

4.5.6 Tableaux d'états

Le Tableau 93 au Tableau 95 définissent le diagramme d'états.

Tableau 93 – Table d'états de SMKPM – initialisation

#	État courant	Événement ou condition =>action	État suivant
6) N1	7) Uninitialized	8) Powerup 9) && OperationalRestore () = "false" 10) => 11) RestoreDefaults ()	12) No Address
13) N2	14) Uninitialized	15) Powerup 16) && OperationalRestore () = "true" 17) => 18) // Pas d'action	19) No Address

Tableau 94 – Table d'états de SMKPM – transitions de réception

#	État courant	Événement ou condition => action	État suivant
20) R1	21) Operational 22) No Tag	23) RcvMsg() = "Any Unconfirmed SM Req Message" 24) && FdaAddressType (sm_svc) = "UNRECOGNIZED" 25) => 26) // Ne rien faire – pas de réponse pour des messages non confirmés	27) Same
28) R2	29) Operational 30) No Tag	31) RcvMsg() = "Any Confirmed SM Req Message" 32) && FdaAddressType (sm_svc) = "UNRECOGNIZED" 33) => 34) SM_ConfirmedService.err { 35) sm_service_type = SvcType (sm_svc), 36) error_class = "access" 37) error_code = "unrecognized FDA Address" 38) addl_code = 2 39) }	40) Same
41) R3	42) Operational 43) No Tag	44) RcvMsg() = "Any Confirmed SM Req Message" 45) && FdaAddressType (sm_svc) = "GROUP SMK" 46) => 47) SM_ConfirmedService.err { 48) sm_service_type = SvcType (sm_svc), 49) error_class = "access" 50) error_code = "unrecognized FDA Address" 51) addl_code = 3 52) }	53) Same
54) R4	55) Operational 56) No Tag	57) RcvMsg() = "Any Confirmed SM Req Message" 58) && IsValid (sm_svc) = "false" 59) => 60) SM_ConfirmedService.err { 61) sm_service_type = SvcType (sm_svc), 62) error_class = "service" 63) error_code = "parameter inconsistent" 64) addl_code = GetAddlCode () 65) }	66) Same
67) R5	68) Operational 69) No Tag	70) RcvMsg() = "Any Confirmed SM Rsp Message" || 71) RcvMsg() = "Any Confirmed SM Error Message" 72) => 73) SM_ConfirmedService.cnf {} * 74) 75) * La façon dont le SMKPM localise l'AP demandeur n'est pas spécifiée. Pour cela, il peut y avoir plusieurs possibilités. La vérification d'erreurs s'effectue au niveau de la réponse par l'AP demandeur.	76) Same
77) R6	78) Operational 79) No Tag	80) RcvMsg() = "SM_FindTagQuery" 81) && IsValid (sm_svc) = "false" 82) => 83) // Ne rien faire – pas de réponse pour des messages non confirmés	84) Same
85) R7	86) Operational	87) RcvMsg() = "SM_FindTagQuery" 88) && (FdaAddressType (sm_svc) = "HSE SMK" 89) || FdaAddressType (sm_svc) = "GROUP SMK") 90) && DeviceRedundancyState () = "secondary" 91) && (QueryType (sm_svc) = "Secondary PD Tag" 92) || QueryType (sm_svc) = "Device Index") 93) && QueryMatch (sm_svc) = "true" 94) => 95) SM_FindTagReply.req {}	96) Same

97) R8	98) Operational	99) RcvMsg() = "SM_FindTagQuery"
		100) && DeviceRedundancyState () = "secondary"
		101) =>
		102) // Écarter le message et ne rien faire. Les secondaires ne répondent pas à
		103) // toute autre interrogation
			104) Same

105) R9	106) Operational	107) RcvMsg() = "SM_FindTagQuery" 108) && (FdaAddressType (sm_svc) = "HSE SMK" 109) || FdaAddressType (sm_svc) = "GROUP SMK") 110) || QueryType (sm_svc) = "Device Index") 111) || QueryType (sm_svc) = "VFD Reference") 112) && QueryMatch (sm_svc) = "true" 113) =>	115) Same

#	État courant	Événement ou condition => action	État suivant
		114) SM_FindTagReply.req {}	
116) R10	117) Operational	118) RcvMsg() = "SM_FindTagQuery" 119) && FdaAddressType (sm_svc) = "HSE SMK" 120) && (QueryType (sm_svc) = “Primary PD Tag” 121) || QueryType (sm_svc) = “VFD Tag”) 122) && QueryMatch (sm_svc) = "true" 123) => 124) SM_FindTagReply.req {}	125) Same
126) R11	127) Operational	128) RcvMsg() = "SM_FindTagQuery" 129) && FdaAddressType (sm_svc) = "GROUP SMK" 130) && (QueryType (sm_svc) = “Primary PD Tag” || 131) QueryType (sm_svc) = “VFD Tag”) 132) && QueryMatch (sm_svc) = "true" 133) => 134) SM_FindTagReply.req {} 135) Type_X_FindTagQuery.ind {}	136) Same
137) R12	138) Operational	139) RcvMsg() = "SM_FindTagQuery" 140) && FdaAddressType (sm_svc) = "GROUP SMK" 141) && (QueryType (sm_svc) = “Primary PD Tag” 142) || QueryType (sm_svc) = “VFD Tag”) 143) && QueryMatch (sm_svc) = "false" 144) => 145) Type_X_FindTagQuery.ind {}	146) Same
147) R13	148) Operational	149) RcvMsg() = "SM_FindTagQuery" 150) && FdaAddressType (sm_svc) = "HSE SMK" 151) && (QueryType (sm_svc) = “Function Block Tag” 152) || QueryType (sm_svc) = "Element ID") 153) => 154) SM_FindTagQuery.ind {}	155) Same
156) R14	157) Operational	158) RcvMsg() = "SM_FindTagQuery" 159) && FdaAddressType (sm_svc) = "GROUP SMK" 160) && (QueryType (sm_svc) = “Function Block Tag” 161) || QueryType (sm_svc) = "Element ID") 162) => 163) SM_FindTagQuery.ind {} 164) Type_X_FindTagQuery.ind {}	165) Same
166) R15	167) Operational	168) RcvMsg() = "SM_FindTagQuery" 169) && (FdaAddressType (sm_svc) = "Type_X SMK" 170) || FdaAddressType (sm_svc) = "LD Type_X INTERFACE SMK" 171) && (QueryType (sm_svc) = “Primary PD Tag” 172) || QueryType (sm_svc) = “Function Block Tag” 173) || QueryType (sm_svc) = "Element ID" 174) || QueryType (sm_svc) = "VFD Tag") 175) => 176) Type_X_FindTagQuery.ind {}	177) Same
178) R16	179) Operational	180) RcvMsg() = "SM_FindTagReply" 181) && FdaAddressType (sm_svc) = "HSE SMK" 182) && DuplicateQueryIdMatch (sm_svc) = "true" 183) && DevId_Match (sm_svc) = "false" 184) => 185) Set_DuplicatePdTagFlag () 186) SM_DeviceAnnunciation.req {} 187) Restart_HseRepeatTimer ()	188) Same
189) R17	190) Operational	191) RcvMsg() = "SM_FindTagReply" 192) && FdaAddressType (sm_svc) = "HSE SMK" 193) && DuplicateQueryIdMatch (sm_svc) = "true" 194) && DevId_Match (sm_svc) = “true” 195) => 196) // Ne rien faire – la réponse provient de cet appareil	197) Same
198) R18	199) Operational	200) RcvMsg() = "SM_FindTagReply" 201) && FdaAddressType (sm_svc) = "HSE SMK" 202) => 203) SM_FindTagReply.ind {} 204) 205) * Livrer l’indication à l’AP ayant émis l'interrogation de Find Tag. Le moyen du SMKPM de localiser l’AP demandeur n'est pas spécifié. Pour cela, il peut y avoir plusieurs possibilités. La vérification d'erreurs s'effectue au niveau de la réponse par l’AP demandeur.	206) Same
207) R19	208) No Tag	209) RcvMsg() = "SM_IdentifyReq" 210) && (FdaAddressType (sm_svc) = "Type_X SMK" 211) || FdaAddressType (sm_svc) = "LD Type_X INTERFACE	219) Same

#	État courant	Événement ou condition => action	État suivant
		SMK") 212) => 213) SM_ConfirmedService.err { 214) sm_service_type = SvcType (sm_svc), 215) error_class = "service" 216) error_code = "object state conflict " 217) addl_code = 19 218) }	
220) R20	221) Operational	222) RcvMsg() = "SM_IdentifyReq" 223) && FdaAddressType (sm_svc) = "Type_X SMK" 224) && SmCacheEntry(sm_svc) = "false" 225) => 226) SM_ConfirmedService.err { 227) sm_service_type = SvcType (sm_svc), 228) error_class = "access" 229) error_code = "object invalidated" 230) addl_code = 20 231) }	232) Same
233) R21	234) Operational 235) No Tag	236) RcvMsg() = "SM_IdentifyReq" 237) => 238) SM_Identify.rsp {}	239) Same
240) R22	241) Operational 242) No Tag	243) (RcvMsg() = "SM_SetAssignmentInfoReq" 244) || RcvMsg() = "SM_ClearAssignmentInfoReq" 245) || RcvMsg() = "SM_ClearAddressReq") 246) && FdaAddressType (sm_svc) = "LD Type_X INTERFACE SMK" 247) => 248) SM_ConfirmedService.err { 249) sm_service_type = SvcType (sm_svc), 250) error_class = "access" 251) error_code = "object access denied" 252) addl_code = 22 253) }	254) Same
255) R23	256) Operational 257) No Tag	258) (RcvMsg() = "SM_SetAssignmentInfoReq" 259) || RcvMsg() = "SM_ClearAssignmentInfoReq" 260) || RcvMsg() = "SM_ClearAddressReq") 261) && DevId_Match (sm_svc) = “false” 262) => 263) SM_ConfirmedService.err { 264) sm_service_type = SvcType (sm_svc), 265) error_class = "service" 266) error_code = “key parameter mismatch” 267) addl_code = GetAddlCode () 268) }	269) Same
270) R24	271) No Tag	272) (RcvMsg() = "SM_SetAssignmentInfoReq" 273) || RcvMsg() = "SM_ClearAssignmentInfoReq" 274) || RcvMsg() = "SM_ClearAddressReq") 275) && FdaAddressType (sm_svc) = "Type_X SMK" 276) => 277) SM_ConfirmedService.err { 278) sm_service_type = SvcType (sm_svc), 279) error_class = "service" 280) error_code = “object state conflict” 281) addl_code = 24 282) }	283) Same
284) R25	285) Operational	286) (RcvMsg() = "SM_SetAssignmentInfoReq" 287) || RcvMsg() = "SM_ClearAssignmentInfoReq" 288) || RcvMsg() = "SM_ClearAddressReq") 289) && FdaAddressType (sm_svc) = "Type_X SMK" 290) && DeviceRedundancyState () = "secondary" 291) => 292) SM_ConfirmedService.err { 293) sm_service_type = SvcType (sm_svc), 294) error_class = "access" 295) error_code = “object access denied” 296) addl_code = 25 297) }	298) Same
299) R26	300) Operational	301) (RcvMsg() = "SM_SetAssignmentInfoReq" 302) || RcvMsg() = "SM_ClearAssignmentInfoReq" 303) || RcvMsg() = "SM_ClearAddressReq") 304) && PdTag_Match() = "false"	312) Same

#	État courant	Événement ou condition => action	État suivant
		305) => 306) SM_ConfirmedService.err { 307) sm_service_type = SvcType (sm_svc), 308) error_class = "service" 309) error_code = “key parameter mismatch” 310) addl_code = GetAddlCode () 311) }	
313) R27	314) Operational	315) RcvMsg() = "SM_SetAssignmentInfoReq" 316) && FdaAddressType (sm_svc) = "Type_X SMK" 317) => 318) Type_X_SetAssignmentInfo.ind{}	319) Same
320) R28	321) Operational 322) No Tag	323) RcvMsg() = "SM_SetAssignmentInfoReq" 324) && PdTagDeviceIndex_Check(sm_svc) = "false" 325) => 326) SM_ConfirmedService.err { 327) sm_service_type = SvcType (sm_svc), 328) error_class = "service" 329) error_code = “parameter inconsistent” 330) addl_code = GetAddlCode () 331) }	332) Same
333) R29	334) No Tag	335) RcvMsg() = "SM_SetAssignmentInfoReq" 336) && FdaAddressType (sm_svc) = "HSE SMK" 337) && DeviceRedundancyState () = “Primary” 338) => 339) Set_Assignment_Data (sm_svc) 340) Clear_DuplicatePdTagFlag () 341) SM_SetAssignmentInfo.rsp {} 342) SM_DeviceAnnunciation.req {} 343) Restart_HseRepeatTimer () 344) SM_FindTagQuery.req {}	345) Operational
346) R30	347) No Tag	348) RcvMsg() = "SM_SetAssignmentInfoReq" 349) && FdaAddressType (sm_svc) = "HSE SMK" 350) && DeviceRedundancyState () = “Secondary” 351) => 352) Set_Assignment_Data (sm_svc) 353) Clear_DuplicatePdTagFlag () 354) SM_SetAssignmentInfo.rsp {} 355) SM_DeviceAnnunciation.req {} 356) Restart_HseRepeatTimer ()	357) Operational
358) R31	359) Operational	360) RcvMsg() = "SM_SetAssignmentInfoReq" 361) && FdaAddressType (sm_svc) = "HSE SMK" 362) => 363) Set_Assignment_Data (sm_svc) 364) SM_SetAssignmentInfo.rsp {} 365) SM_DeviceAnnunciation.req {} 366) Restart_HseRepeatTimer ()	367) Same
368) R32	369) No Tag	370) RcvMsg() = "SM_ClearAssignmentInfoReq" 371) && FdaAddressType (sm_svc) = "HSE SMK" 372) => 373) // Cela cause le retour de l'appareil vers les valeurs par défaut. 374) RestoreDefaults () 375) SM_ClearAssignmentInfo.rsp {} 376) SM_DeviceAnnunciation.req {} 377) Restart_HseRepeatTimer ()	378) Same
379) R33	380) Operational	381) RcvMsg() = "SM_ClearAssignmentInfoReq" 382) && FdaAddressType (sm_svc) = "Type_X SMK" 383) => 384) Type_X_ClearAssignmentInfo.ind{}	385) Same
386) R34	387) Operational	388) RcvMsg() = "SM_ClearAssignmentInfoReq" 389) && FdaAddressType (sm_svc) = "HSE SMK" 390) => 391) RestoreDefaults () 392) SM_ClearAssignmentInfo.rsp {} 393) SM_DeviceAnnunciation.req {} 394) Restart_HseRepeatTimer ()	395) No Tag
396) R35	397) Operational	398) RcvMsg() = "SM_ClearAddressReq" 399) && FdaAddressType (sm_svc) = "Type_X SMK" 400) => 401) Type_X_ClearAddress.ind{}	402) Same
403)	404) Operational	406) RcvMsg() = "SM_ClearAddressReq"	417) Same

#	État courant	Événement ou condition => action	État suivant
R36	405) No Tag	407) && FdaAddressType (sm_svc) = "HSE SMK" 408) && (AddressToClear(sm_svc) = "None" 409) || ConfigurationSessionActive = “True”) 410) => 411) SM_ConfirmedService.err { 412) sm_service_type = SvcType (sm_svc), 413) error_class = "service" 414) error_code = “object constraint conflict” 415) addl_code = GetAddlCode () 416) }	
418) R37	419) Operational 420) No Tag	421) RcvMsg() = "SM_ClearAddressReq" 422) && FdaAddressType (sm_svc) = "HSE SMK" 423) && (NumberOfAssignedAddresses() = 1 424) || AddressToClear(sm_svc) = "All") 425) => 426) SM_ClearAddress.rsp {} 427) Clear_Address (interface_to_clear)	428) No Address
429) R38	430) Operational 431) No Tag	432) RcvMsg() = "SM_ClearAddressReq" 433) && FdaAddressType (sm_svc) = "HSE SMK" 434) && NumberOfAssignedAddresses() > 1 435) => 436) Clear_Address (interface_to_clear) 437) SM_ClearAddress.rsp {} 438) SM_DeviceAnnunciation.req {} 439) Restart_HseRepeatTimer ()	440) Same

Tableau 95 – Table d'états de SMKPM – événements internes

#	État courant	Événement ou condition => action	État suivant
441) S1	442) Operatio nal 443) No Tag	444) HseRepeatTimerExpires() 445) => 446) SM_DeviceAnnunciation.req {} 447) Restart_HseRepeatTimer ()	448) Same
449) S2	450) No Address	451) RcvNewNetworkAddress (interface, address) 452) && AssignmentInfo_Set () = "false" 453) => 454) RestoreDefaults () 455) NewAddress (interface, address) 456) SM_DeviceAnnunciation.req {} 457) Restart_HseRepeatTimer ()	458) No Tag
459) S3	460) No Address	461) RcvNewNetworkAddress (interface, address) 462) && AssignmentInfo_Set () = "true" 463) && DeviceRedundancyState () = “Primary” 464) => 465) Clear_DuplicatePdTagFlag () 466) NewAddress (interface, address) 467) SM_DeviceAnnunciation.req {} 468) Restart_HseRepeatTimer () 469) SM_FindTagQuery.req {}	470) Operati onal
471) S4	472) No Address	473) RcvNewNetworkAddress (interface, address) 474) && AssignmentInfo_Set () = "true" 475) && DeviceRedundancyState () = “Secondary” 476) => 477) Clear_DuplicatePdTagFlag () 478) NewAddress (interface, address) 479) SM_DeviceAnnunciation.req {} 480) Restart_HseRepeatTimer ()	481) Operati onal

482)
S5
	483) Operatio
	nal
		484) // pour une nouvelle adresse, une seconde interface HSE ou pour le changement d'adresse
		485) // interface HSE
		486) RcvNewNetworkAddress (interface, address)
		487) && NetworkAddressChange (interface, address) = “false”
		488) =>
		489) NewAddress (interface, address)
		490) SM_DeviceAnnunciation.req {}
		491) Restart_HseRepeatTimer ()
			492) Same

493) S6	494) Operatio nal	495) SM_UnconfirmedService.req {} 496) || SM_ConfirmedService.req {} 497) || SM_ConfirmedService.rsp {} 498) || Type_X_ConfirmedService.rsp {} 499) => 500) Send_SM_ReqRspMessage (sm_svc)	501) Same
502) S7	503) Operatio nal	504) SM_ConfirmedService.err{} 505) || Type_X_ConfirmedService.err {} 506) => 507) Send_SM_CommonErrorRsp (508) sm_service_type = SvcType (sm_svc), 509) error_class = "result error class" 510) error_code = “result error code” 511) addl_code = “result error addl_code” 512) addl_description = “result error addl_description”) 513))	514) Same
515) S8	516) Operatio nal 517) No Tag	518) SntpSyncLost() 519) => 520) SM_DeviceAnnunciation.req {} 521) Restart_HseRepeatTimer ()	522) Same

4.5.7 Descriptions des fonctions

4.5.7.1 Généralités

Les descriptions suivantes définissent les fonctions référencées dans les tables d'états précédentes.

4.5.7.2 Fonctions de l'événement

4.5.7.2.1 Résumé

Ce sont les fonctions de *rappel* utilisées pour la réception d'un événement. Elles ne génèrent pas les modifications des données reçues. Les autres événements sont représentés par les Primitives de Services définies précédemment. Les définitions des fonctions sont représentées dans les Tableaux 96 à 99.

4.5.7.2.2 HseRepeatTimerExpires ()

Tableau 96 – HseRepeatTimerExpires ()

Fonction	
Cette fonction est invoquée quand Hse Repeat Time arrive à expiration.	
Paramètres d'entrée	Paramètres de sortie
None	None

4.5.7.2.3 RcvNewNetworkAddress (interface, adresse)

Tableau 97 – RcvNewNetworkAddress (interface, adresse)

Fonction	
Cette fonction est invoquée quand une adresse de réseau est reçue à partir des couches sous-jacentes. Les paramètres d'entrée identifient l'interface de Type 5 et l'adresse de réseau reçue.	
Paramètres d'entrée	Paramètres de sortie
interface, address	None

4.5.7.2.4 RcvMsg ()

Tableau 98 – RcvMsg ()

Fonction	
Cette fonction est invoquée après la réception d'une APDU. Elle achemine le paramètre sm_svc vers le diagramme d'états. Ce paramètre contient tous les champs des APDU, y compris l'en-tête et les champs de fin de trame. RcvMsg() = "SM_XXX' de syntaxe est utilisée pour indiquer la réception d'une APDU pour le service SM XXX.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	None

4.5.7.2.5 SntpSyncLost ()

Tableau 99 – SntpSyncLost ()

Fonction	
Cette fonction est invoquée lorsque l'état de synchronisation de temps entre l'appareil et le serveur de temps distant sélectionné passe de "true" à "false".	
Paramètres d'entrée	Paramètres de sortie
none	none

4.5.7.3 Fonctions de condition

4.5.7.3.1 Résumé

Ces fonctions sont utilisées pour tester un événement. Elles ne génèrent pas les modifications des données testées. Les définitions des fonctions sont représentées dans les Tableaux 100 à 116.

4.5.7.3.2 AddressToClear (sm_svc)

Tableau 100 – AddressToClear (sm_svc)

Fonction	
Renvoie les interfaces de Type 5 qui sont en cours de demande de suppression. Les valeurs impliquées sont des valeurs OperationalnetworkAddress, Non-OperationalNetworkAddress, All, et None. "None" indique que l'interface de Type 5 spécifiée n'a pas d'adresse ou n'est pas capable de demander une nouvelle adresse de façon dynamique.	
Paramètres d'entrée	Paramètres de sortie
none	HseInterface

4.5.7.3.3 AssignmentInfo_Set ()

Tableau 101 – AssignmentInfo_Set()

Fonction	
Renvoie "true" si les informations d'attribution de SMK ont été établies précédemment par le service Set Assignment et sont encore valides telles que définies par ce service.	
Paramètres d'entrée	Paramètres de sortie
none	true ou false

4.5.7.3.4 ConfigurationSessionActive ()

Tableau 102 – ConfigurationSessionActive ()

Fonction	
Renvoie "true" à condition qu'une AR de configuration soit établie.	
Paramètres d'entrée	Paramètres de sortie
none	true ou false

4.5.7.3.5 DeviceRedundancyState ()

Tableau 103 – DeviceRedundancyState ()

Fonction	
Remet l'appareil dans son état de redondance.	
Paramètres d'entrée	Paramètres de sortie
none	primary ou secondary

4.5.7.3.6 DevId_Match (sm_svc)

Tableau 104 – DevId_Match (sm_svc)

Fonction	
Renvoie "true" si Device ID dans une APDU correspond exactement à Device ID de ce service.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.3.7 DuplicateQueryIdMatch (sm_svc)

Tableau 105 – DuplicateQueryIdMatch (sm_svc)

Fonction	
Renvoie "true" si Invoke Id correspond à Invoke Id de la demande de Find Tag qui a été envoyée pour déterminer si un autre appareil possède le même PD Tag que cet appareil.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.3.8 DuplicatePdTagDetected ()

Tableau 106 – DuplicatePdTagDetected ()

Fonction	
Renvoie "true" si l'appareil a détecté un autre appareil avec le même Physical Device Tag que ce service.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.3.9 FdaAddressType (sm_svc)

Tableau 107 – FdaAddressType (sm_svc)

Fonction	
Évalue le composant de FDA Address du paramètre sm_svc et renvoie le SMK qu'il l'identifie. Pour les valeurs des renvois définis ci-dessous, LL LL identifie une liaison de Type "X" connectée à l'appareil de liaison, alors que NN représente l'adresse de nœud pour cette interface de liaison. Renvois : SMK de Type 5 si FDA Address = 00 00.00.SM, où SM est le sélecteur de SMK GROUP SMK si FDA Address = 00 00 GG GG ou LL.GG, où GG GG est l'adresse de groupe de SMK de Type "X". LD Type "X" Interface SMK si FDA Address = LL LL NN.SM et NN correspond à l'adresse de nœud d'une interface LD de Type "X". Type "X" SMK si FDA Address = LL LL.NN.SM UNRECOGNIZED si FDA Address ne correspond pas aux descriptions ci-dessus.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	SMK Identifier

4.5.7.3.10 IsValid (sm_svc)

Tableau 108 – IsValid (sm_svc)

Fonction	
Renvoie "true" si les paramètres dans une APDU sont valides pour chaque type de service. Les données sont considérées valides, si elles possèdent le même type de données et si elles sont conformes aux valeurs définies pour elles. Les valeurs du paramètre sont définies dans les spécifications du service de SMK dans la CEI 61158-5 et dans les spécifications des APDU de la présente spécification. La validité des valeurs est vérifiée pour cette fonction. Prendre en compte le contraste avec d'autres fonctions de condition définies pour le SMKPM, telles que PdTag_Match(sm_svc) qui vérifient l'utilisation de la valeur.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.3.11 NetworkAddressChange (interface, adresse)

Tableau 109 – NetworkAddressChange (interface, adresse)

Fonction	
Renvoie "true" si l'interface spécifiée possède déjà une adresse de réseau valide.	
Paramètres d'entrée	Paramètres de sortie
interface, address	true ou false

4.5.7.3.12 NumberOfAssignedAddresses ()

Tableau 110 – NumberOfAssignedAddresses ()

Fonction	
Renvoie le numéro des interfaces de Type 5 qui possèdent actuellement une adresse de réseau.	
Paramètres d'entrée	Paramètres de sortie
None	NumberOfInterfaces

4.5.7.3.13 OperationalRestore ()

Tableau 111 – OperationalRestore ()

Fonction	
Renvoie "true" si Operational Powerup est "true" et le dernier état de la SMKPM était "Operational".	
Paramètres d'entrée	Paramètres de sortie
None	true ou false

4.5.7.3.14 PdTag_Match (sm_svc)

Tableau 112 – PdTag_Match (sm_svc)

Fonction	
Renvoie "true" si PD Tag dans une APDU correspond à PD Tag de l'appareil identifié par FDA Address de l'APDU. Le renvoi de "false" est généré par les adresses de FDA de groupe.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.3.15 PdTagDeviceIndex_Check (sm_svc)

Tableau 113 – PdTagDeviceIndex_Check (sm_svc)

Fonction	
Renvoie "false" si : 1. le service pouvait générer la suppression de PD Tag et Device Index (les deux doivent être établis), soit 2. PD Tag dans l'appareil est établi avant que le service commence; le service tente de le modifier (Device Index peut être modifié, mais pas PD Tag). Sinon, renvoie "true".	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.3.16 Query_Match (sm_svc)

Tableau 114 – Query_Match (sm_svc)

Fonction	
Renvoie "true" si l'appareil contient l'objet interrogé.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.3.17 QueryType (sm_svc)

Tableau 115 – QueryTyoe (sm_svc)

Fonction	
Renvoie Query Type pour Find Tag Query ou Find Tag Reply.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	query type

4.5.7.3.18 SmCacheEntry (sm_svc)

Tableau 116 – SmCacheEntry (sm_svc)

Fonction	
Renvoie "true" si le SMK a saisi des informations d'identification pour l'appareil de Type "X" identifié par le composant fda_address du paramètre sm_svc.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	true ou false

4.5.7.4 Fonctions d'action

4.5.7.4.1 Résumé

Ces fonctions sont utilisées dans la portion Action des transitions. Les définitions des fonctions sont représentées dans les Tableaux 117 à 129.

4.5.7.4.2 Clear_Address (interface_to_clear)

Tableau 117 – Clear_Address (interface_to_clear)

Fonction	
Fermer toutes les sessions de FDA et les VCR de Type 9 pour supprimer l'adresse. Arrêter Repeat Timer de Type 5, s'il est en cours d'opération. Supprimer l'adresse pour l'interface de Type 5 spécifiée dans le tableau d'adresses du réseau local. Supprimer l'adresse par le biais des couches sous-jacentes. Si Operational network address n'est plus valide, l'établir à l'adresse de l'autre interface de Type 5, si cette adresse est valide.	
Paramètres d'entrée	Paramètres de sortie
interface_to_clear	None

4.5.7.4.3 Clear_DuplicatePdTagFlag ()

Tableau 118 – Clear_DuplicatePdTagFlag ()

Fonction	
Exécute l'équivalent de DuplicateDetectedState :=DuplicateDetectedState et NON “DuplicatePdTag”	
Paramètres d'entrée	Paramètres de sortie
none	None

4.5.7.4.4 Get_AddlCode ()

Tableau 119 – Get_AddlCode ()

Fonction	
Détermine la valeur de la variable des codes supplémentaires. Ces valeurs sont définies en 4.5.6	
Paramètres d'entrée	Paramètres de sortie
none	addl_code

4.5.7.4.5 New_Address (interface, adresse)

Tableau 120 – New_Address (interface, adresse)

Fonction	
Si l'adresse de réseau existant pour l'interface de Type 5 spécifiée est valide, fermer toutes les sessions de FDA client/serveur et les VCR de Type 9 pour cette adresse. Établir ou modifier l'adresse pour l'interface de Type 5 spécifiée dans le tableau d'adresses du réseau local. Si Operational network address n'est pas valide ou n'est plus valide, l'établir à cette adresse. Si Operational Network Address était valide ou a été modifiée, déplacer immédiatement tous les VCR source d'alertes et éditeur et les Sessions de FDA existants vers Operational NetworkAddress.	
Paramètres d'entrée	Paramètres de sortie
interface, address	None

4.5.7.4.6 Restart_HseRepeatTimer ()

Tableau 121 – Restart_HseRepeatTimer ()

Fonction	
Cette fonction arrête HseRepeatTimer, s'il est en cours d'opération, et recommence avec une période de temps définie par l'attribut Repeat Time de Type 5. Quand le chronomètre expire, il génère l'événement de HseRepatTimerExpires pour le diagramme d'états de SM.	
Paramètres d'entrée	Paramètres de sortie
None	None

4.5.7.4.7 Restore_Defaults ()

Tableau 122 – Restore_Defaults ()

Fonction	
Restaurer l'appareil sur sa condition 'as manufactered', sauf si les adresses de réseau et Operational network address sont gardées en mémoire. Cela ferme toutes les sessions et les VCR et supprime également toutes les données opérationnelles configurées.	
Paramètres d'entrée	Paramètres de sortie
None	None

4.5.7.4.8 Send_SM_CommonErrorRsp (sm_service_type, svc_spec_params)

Tableau 123 – Send_SM_CommonErrorRsp (sm_service_type, svc_spec_params)

Fonction	
Construit et envoie les APDU de réponse d'erreurs pour le service identifié dans le paramètre sm_service_type.	
Paramètres d'entrée	Paramètres de sortie
sm_service_type, svc_spec_params	None

4.5.7.4.9 Send_SM_ReqRspMessage (sm_svc)

Tableau 124 – Send_Sm_ReqRspMessage (sm_svc)

Fonction	
Construit et envoie les APDU de demande et de réponse pour le service identifié dans le paramètre sm_service_type.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	None

4.5.7.4.10 Set_Assignment_Data (sm_svc)

Tableau 125 – Set_Assignment_Data (sm_svc)

Fonction	
Copie les données d'attribution pour un appareil de Type 5 des APDU vers les attributs de SMK correspondants. Si Operational IP Address change, déplacer immédiatement tous les VCR source d'alertes et éditeur existants et les Sessions de FDA vers Operational IP Address.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	none

4.5.7.4.11 Set_DuplicatePdTagFlag ()

Tableau 126 – Set_DuplicatePdTagFlag ()

Fonction	
Établit l'attribut DuplicateDetectedState pour indiquer que cet appareil et un autre appareil possèdent le même PD Tag.	
Paramètres d'entrée	Paramètres de sortie
None	none

4.5.7.4.12 SvcType (sm_svc)

Tableau 127 – SvcType (sm_svc)

Fonction	
Cette fonction décode les données qui sont acheminées dans le paramètre sm_svc et renvoie le type de service.	
Paramètres d'entrée	Paramètres de sortie
sm_svc	service type

4.5.8 Codes et classes d'erreurs

L'application des codes et des classes d'erreurs définis est représentée en 4.3.6.4.

4.5.9 Définition du code supplémentaire

Le Tableau 697 et le Tableau 698 définissent les valeurs qui peuvent être utilisées pour un octet de code supplémentaire des classes d'erreurs typiques de FDA. Cette définition est propre aux services de SM.

Tableau 128 – Code supplémentaire utilisé par les codes et classes d'erreurs

Classe d'erreur de FDA	Code d'erreur de FDA	Valeur du code supplémentaire pour SMKPM	Fourni par
service	key parameter mismatch	Voir Tableau 129	GetAddlCode ()
	parameter inconsistent	Voir Tableau 129	GetAddlCode ()
	object constraint conflict	Voir Tableau 129.	GetAddlCode ()

Tableau 129 – ID des paramètres des codes supplémentaires

Paramètre	Code supplémentaire
Clear Duplicate Detection State	5
Device ID	10
Device Index	20
Device Redundancy State	25
Element ID	30
FDA Address	35
Function Block Tag	40
Type "X" New Address	45
Interface to Clear	50
Invoke ID	55
PD Tag	60

Paramètre	Code supplémentaire
Type 5 Repeat Time	65
LAN Redundancy Addr	70
Max Device Index	75
Operational IP Address	80
Query Type	85
VFD Reference	90
VFD Tag	95
Interface n'a pas d'adresse	101
Interface a une adresse fixe	102
Interface a une session de configuration	103

4.6 Diagramme d'états de VCR

Le diagramme d'états de VCR de Type 5 est défini par le diagramme d'états de VCR de Type 9 avec les modifications suivantes.

- Le service Initiate de Type 5 remplace le service Initiate de Type 9. Les paramètres du service Initiate de Type 9, indéfinis pour le service Initiate de Type 5, sont ignorés dans le diagramme d'états.
- Les références au service AR Abort par les transitions d'états "Connection Establishment" de Type 9 sont modifiées comme suit. L'envoi d'une APDU de demande de VCR Abort de Type 5 remplace la primitive "request" de Abort, et la réception d'une APDU de demande de VCR Abort de Type 5 remplace la primitive "indication" de Abort de Type 9. La seule exception à cette règle est lorsqu'une primitive "Abort.request" de Type 9 est émise en réponse à la réception d'une PDU de demande de Initiate de Type 9 au moment où le VCR est en état de connexion établie. Dans ce cas, l'Agent de FDA renvoie une APDU de Initiate Error de Type 6 avec la même classe d'erreur et le même code d'erreur fournis dans la primitive "Abort.request" de Type 9.
- Les points d'extrémité de VCR Client et Serveur de Type 5 utilisent les APDU de Idle de Type 5 en tant que APDU "keep alive" comme suit.
- Les points d'extrémité VCR client de Type 5 envoient les APDU de demande de Idle seulement quand le chronomètre Idle arrive à expiration.
- Les points d'extrémité de VCR serveur de Type 5 envoient immédiatement les APDU de réponse de Idle quand ils reçoivent les APDU de demande de Idle.
- Les points d'extrémité de client relancent leurs chronomètres de Idle toutes les fois qu'ils envoient une APDU de Type 5.
- Les points d'extrémité de serveur ne prennent pas en charge le chronomètre Idle.
- A l'expiration de leur chronomètre d'inactivité, les points d'extrémité de VCR client et serveur de Type 5 s'arrêtent après avoir envoyé une APDU de Abort.
- Les demandes d'établir un VCR de Type 5 à l'instar de "Type 9 CLIENT", en utilisant l'option du Sélecteur de VCR, sont rejetées si l'adresse du sélecteur distant de VCR Client de Type 9 indique que le VFD distant est représenté par MIB VFD. La classe d'erreur utilisée est "access" et le code d'erreur est "object access denied".
- Les points d'extrémité de serveur, qui sont établis pour l'accès MIB, répondent négativement aux demandes de service de mises à jour de Type 9 en utilisant la classe d'erreur "access" et le code d'erreur "object access denied", si l'AR sous-jacente n'est pas une AR de configuration. Les demandes de service de mises à jour de Type 9 :
  - 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

4.7 Machine protocolaire de service de FAL (FSPM)

4.7.1 FSPM de Type 5

L'interface formelle entre un VCR de Type 5 et son AR sous-jacente assure la compatibilité avec les transitions d'état de VSR de Type 9 utilisées pour le diagramme d'états de VCR de Type 5.

La FSPM de Type 5 est définie par la FSPM de Type 9. Elle est utilisée dans son intégralité avec les modifications suivantes :

- Les primitives des services Confirmed Send et Unconfirmed Send pour l'envoi et la réception sont étendues pour pouvoir contenir le paramètre Type 5_VCR_ID et identifier ainsi le VCR qui utilise le point d'extrémité de la session.
- Toutes les références aux adresses de DLCEP distantes sont remplacées par les références aux adresses distantes. Utilisé avec des AR fonctionnant sur les connexions, le paramètre d'adresse distante identifie la connexion.
- Le service Abort n'est pas acheminé entre la FSPM et les diagrammes d'états de VCR de Type 5. En revanche, une APDU de VCR Abort de Type 5 est acheminée entre les deux diagrammes d'états à l'aide du service Unconfirmed Send.
- Il existe certaines différences entre les paramètres de services dans les primitives de service échangées entre la FSPM de Type 5 et l'ARPM de Type 5. Les primitives des services et leurs paramètres sont définis en 4.8.1.1.1.

4.8 Machine protocolaire des relations entre applications (ARPM)

4.8.1 ARPM de la session client / serveur

4.8.1.1 Définitions des primitives

4.8.1.1.1 Primitives échangées entre ARPM et FSPM

Les Tableaux 130 et 131 définissent les primitives échangées entre l'ARPM et la FSPM.

Tableau 130 – Primitives émises par la FSPM vers l'ARPM

Nom de la primitive	Source	Paramètres associés	Fonctions
EST_req	FSPM	arep_id, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish request" de la FSPM vers l'ARPM.
EST_rsp(+)	FSPM	arep_id, user_data,	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish response(+)" de la FSPM vers l'ARPM.
EST_rsp(-)	FSPM	arep_id, user_data, service_type	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish response(-)" de la FSPM vers l'ARPM.
Abort_req	FSPM	vcr_id, abort_detail, abort_identifier, reason_code	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Abort request" de la FSPM vers l'ARPM.
CS_req	FSPM	vcr_id, service_type, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive " Confirmed Send (CS) request" de la FSPM vers l'ARPM.
CS_rsp	FSPM	vcr_id, service_type, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive " Confirmed Send (CS) response" la FSPM vers l'ARPM.
523) UCS_req	524) FSPM	525) vcr_id, 526) service_type, 527) user_data	528) C'est une primitive interne de FAL utilisée pour acheminer une primitive " Unconfirmed Send (CS) request" de la FSPM vers l'ARPM.

Tableau 131 – Primitives émises par ARPM vers FSPM

Nom de la primitive	Source	Paramètres associés	Fonctions
EST_ind	ARPM	arep_id, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish indication" de la FSPM vers l'ARPM.
EST_cnf(+)	ARPM	arep_id, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish confirmation(+)" de la FSPM vers l'ARPM.
EST_cnf(-)	ARPM	arep_id, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish confirmation(-)" de la FSPM vers l'ARPM.
Abort_ind	ARPM	vcr_id, abort_detail, abort_identifier, reason_code	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Abort" de la FSPM vers l'ARPM.
CS_ind	ARPM	vcr_id, service_type, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Confirmed Send (CS) indication" de la FSPM vers l'ARPM.
CS_cnf(+)	ARPM	vcr_id, service_type, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive " Confirmed Send (CS) confirmation(+)" de la FSPM vers l'ARPM.
CS_cnf(-)	ARPM	vcr_id, service_type, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive " Confirmed Send (CS) confirmation(-)" de la FSPM vers l'ARPM.
529) UCS_ind	530) ARPM	531) vcr_id, 532) service_type, 533) user_data	534) C'est une primitive interne de FAL utilisée pour acheminer une primitive "Unconfirmed Send (CS)" de la FSPM vers l'ARPM.

4.8.1.1.2 Paramètres des primitives de FSPM/ARPM

Les paramètres utilisés avec les primitives échangées entre FSPM et ARPM sont décrits dans le Tableau 132.

Tableau 132 – Paramètres utilisés avec les primitives échangées entre FSPM et ARPM

Nom du paramètre	Description
arep_id	Ce paramètre est utilisé pour l'identification non ambiguë d'une instance d'AREP ayant émis une primitive. Le moyen d'une telle identification n'est pas spécifié par la présente norme.
user_data	Ce paramètre achemine les données d'utilisateur de FAL.
abort_identifier	Ce paramètre achemine la valeur utilisée pour le paramètre Abort_Identifier.
reason_code	Ce paramètre achemine la valeur utilisée pour le paramètre Reason_Code.
abort_detail	Ce paramètre achemine la valeur utilisée pour le paramètre Abort_Detail.
vcr_id	Ce paramètre est utilisé pour l'identification non ambiguë d'une instance de VCR ayant émis une primitive ou la destination de la primitive. Le moyen d'une telle identification n'est pas spécifié par la présente norme.
535) service_type	536) Ce paramètre achemine le protocole Id et le service dans les limites du protocole.

4.8.1.2 Mapping de DLL de la Classe AREP Client / Serveur

4.8.1.2.1 Définition formelle

Le Paragraphe 4.8.1.2 décrit le mapping de la classe AREP Client / Serveur sur le modèle Socket. Le Paragraphe 4.8.1.2 définit les attributs du mapping de DLL et leurs valeurs autorisées et utilisées avec la classe AREP Client / Serveur.

CLASS : ClientServer
PARENT CLASS : Point d'extrémité d'AR
ATTRIBUTES

1	(m)	KeyAttribute (Attribut-clé):	LocalAndRemoteAddresses
1.1	(m)	KeyAttribute:	LocalAddress
1.2	(m)	KeyAttribute:	RemoteAddress
2	(m)	Attribut:	TransferType (CONN, CNLS)
3	(m)	Attribut	BufferSize
4	(m)	Attribut:	TransmitDelayTime
5	(m)	Attribut:	InactivityCloseTime
6	(m)	Attribut:	HdrOptions
7	(m)	Attribut:	GuardBand
8	(m)	Attribut:	MibConfigurationUse
9	(m)	Attribut:	MaxApduLength
10	(m)	Attribut:	MaxOutstanding
11	(m)	Attribut:	ServerPhysicalDeviceTag

4.8.1.2.2 Attributs

LocalAndRemoteAddresses

Cet attribut clé identifie l'AREP. Il est composé des adresses distantes et locales de la couche sous-jacente pour l'AREP.

TransferType

Cet attribut spécifie si les services orientés connexion (CONN) ou sans connexion (CNLS) des couches sous-jacentes sont utilisés par l'AREP.

BufferSize

Cet attribut spécifie la taille en octets du tampon d'envoi du point d'extrémité de la session. Le tampon contient un nombre entier des APDU. S'il est reçu une APDU pour le transfert ferait déborder le tampon, c'est le tampon qui est envoyé en premier et, ensuite, l'APDU est ajoutée comme étant la première APDU dans le tampon.

TransmitDelayTime

Cet attribut spécifie la quantité de temps accumulée par un point d'extrémité de la session qui envoie des APDU dans son tampon. La période de temps commence lorsque la première APDU est placée dans le tampon émetteur soit après l'initialisation du tampon, soit après avoir été vidé par la transmission précédente.

Si le point d'extrémité n'a pas d'APDU à envoyer (le tampon est vide), mais il reçoit une APDU à envoyer, il la charge dans le tampon et attend des APDU supplémentaires à enchaîner dans le tampon. Si le tampon se remplit avant que Transmit Delay Time soit atteint, La FAL AE de Type 5 arrête le chronomètre et envoie le tampon. Lorsque cet intervalle de temps a expiré et le tampon n'est pas encore plein, il envoie le contenu du tampon.

InactivityCloseTime

Cet attribut est utilisé par les points d'extrémité Client et Serveur pour déterminer le moment de la fermeture de ses points d'extrémité pour cause d'inactivité.

HdrOptions

Cet attribut spécifie les champs d'en-tête des APDU qui sont présents dans les APDU envoyées et reçues sur la session.

GuardBand

Cet attribut spécifie la limite inférieure et supérieure pour le champ protégé du repassage par zéro («rollover»).

MibConfigurationUse

Cet attribut Boolean spécifie, à condition d'être "TRUE" (VRAI), que l'AR est utilisée pour une configuration de MIB. Ce n'est pas un attribut qui peut être configuré. Les applications clientes sont censées savoir qu'elles ont besoin des AR de configuration et les demander par la demande Establish.

MaxApduLength

Cet attribut spécifie la longueur maximale autorisée en octets des APDU envoyées par les AREP de CLIENT et reçues par les AREP de SERVEUR.

MaxOutstanding

Cet attribut spécifie le nombre maximal de demandes que le point d'extrémité destinataire peut avoir en cours à tout moment. "Outstanding" signifie que le point d'extrémité a reçu un message de demande, mais qu'il n'a pas renvoyé le message de réponse correspondant. Sa valeur initiale est établie par le fabricant de l'appareil. La réception d'un message de demande génère le dépassement de cet attribut. Une réponse négative est renvoyée avec une classe d'erreur = "resource" et un code d'erreur = "max outstanding requests per session exceeded.

ServerPhysicalDeviceTag

Cet attribut spécifie SMK PhysicalDeviceTag du point d'extrémité de serveur.

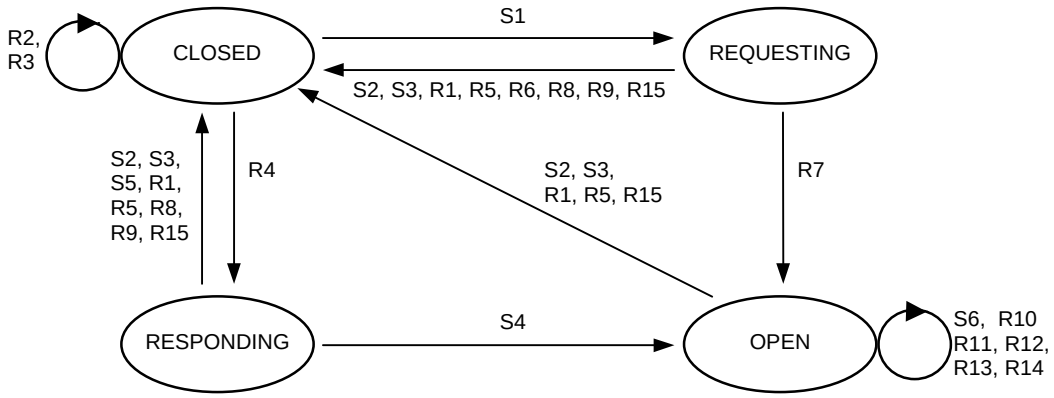
4.8.1.3 Diagramme d'états d'AREP Client / Serveur

4.8.1.3.1 Les états d'ARPM Client / Serveur

Les états définis et leurs descriptions de l'ARPM Client / Serveur sont représentés dans le Tableau 133 et la Figure 2. Si le type de transfert = CONN, l'AR fonctionne sur une connexion sous-jacente, et le diagramme d'états ne prend pas son effet jusqu'à ce que la connexion sous-jacente soit établie. La connexion sous-jacente est terminée, si le premier message, qui a été reçu au niveau de la connexion sous-jacente, n'est pas une demande d'établissement.

Tableau 133 – États d'ARPM Client / Serveur

États	Description
CLOSED (FERMÉ)	Des AREP de Client et de Serveur doivent être crées avant l'utilisation. Ils sont créés dans un état CLOSED (fermé). Quand un Agent de FDA reçoit une FDA-PDU de demande de OpenSession, il crée de façon dynamique un nouveau serveur d'AREP et livre la FDA-PDU reçue. Les AREP dans l'état CLOSED ne sont pas capables d'envoyer ou de recevoir des FDA-PDU à l'exception des FDA-PDU de OpenSession. Avant d'être supprimé, la transition d'un AREP vers l'état CLOSED provoque l'arrêt de ses chronomètres et la fermeture de sa connexion socket sous-jacente (si nécessaire).
OPEN (OUVERT)	L'AREP est défini et est capable d'envoyer ou de recevoir des FAL-PDU.
REQUESTING (DEMANDE EN COURS)	L'AREP a envoyé une FAL-PDU de demande d'établissement et attend une réponse en provenance de l'AREP distant.
RESPONDING (RÉPONSE EN COURS)	L'AREP a reçu une FAL-PDU de demande d'établissement, a livré une primitive "Establish.ind" et attend une réponse en provenance de son utilisateur.
537) SAME (INCHANGÉ)	538) Indique que l'état suivant est le même que l'État courant.



Légende

Anglais	Français
CLOSED	Fermé
REQUESTING	En demande
RESPONDING	En réponse
OPEN	Ouvert

Figure 2 – Diagramme de transition d'états d'ARPM client / serveur

4.8.1.3.2 Table d'états d'ARPM client / serveur

Les Tableaux 134 et 135 définissent le diagramme d'états d'ARPM Client / Serveur.

Tableau 134 – Table d'états d'ARPM Client / Serveur – transitions d'expéditeur

#	État courant	Événement ou condition => action	État suivant
539) S1	540) CLOSED	541) EST_req 542) && EndpointType = "CLIENT" 543) => 544) FAL-PDU_req { 545) smpm_service_name = "SMPM_Send", 546) arep_id = GetArepId (), remote_address = RemoteAddress, 547) fda_pdu = BuildFAL-ReqRspPDU (service_type = "AR Establish Req", 548) fal_data = user_data) 549) }, 550) 551) StartInactivityCloseTimer()	552) REQUESTING
553) S2	554) RESPONDING 555) REQUESTING 556) OPEN	557) Abort_req 558) =>	559) CLOSED
560) S3	561) REQUESTING RESPONDING 562) OPEN	563) InactivityCloseTimerExpires 564) =>	565) CLOSED
566) S4	567) RESPONDING	568) EST_rsp(+) 569) => 570) FAL-PDU_req { 571) smpm_service_name = "SMPM_Send", 572) arep_id = GetArepId (), remote_address = RemoteAddress, 573) fda_pdu = BuildFAL-ReqRspPDU (service_type = "AR Establish Rsp", 574) fal_data = user_data) 575) } 576) 577) StartInactivityCloseTimer()	578) OPEN
579) S5	580) RESPONDING	581) EST_rsp(-) 582) => 583) FAL-PDU_req { 584) smpm_service_name = "SMPM_Send", 585) arep_id = GetArepId (), remote_address = RemoteAddress, 586) fda_pdu = BuildFAL-ReqRspPDU (service_type = "AR Establish Err", 587) fal_data = user_data) 588) }	589) CLOSED
590) S6	591) OPEN	592) UCS_req 593) || CS_req 594) || CS_rsp 595) => 596) FAL-PDU_req { 597) smpm_service_name = "SMPM_Send", 598) arep_id = GetArepId (), remote_address = RemoteAddress, 599) fda_pdu = BuildFAL-ReqRspPDU (service_type = service_type, 600) fal_data = user_data) 601) }	602) OPEN

Tableau 135 – Table d'états ARPM client / serveur – transitions de destinataire

#	État courant	Événement ou condition => action	État suivant
603) R1	604) REQUESTING 605) RESPONDING OPEN	606) FAL-PDU_ind 607) && FDA_PDU_Valid(fal_pdu) = False 608) => 609) FAL-PDU_req* { 610) smpm_service_name = "SMPM_Send", 611) arep_id = GetArepId (), 612) remote_address = FAL_Pdu_RemoteAddress(fal_pdu), 613) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 614) error_class = "service", 615) error_code = appropriate error code for detected error) 616) } 617) 618) * Utilisé seulement quand la PDU de FAL reçue représente une demande de DTC	619) CLOSED
620) R2	621) CLOSED	622) FAL-PDU_ind 623) && EndpointType = "SERVER" 624) && FAL_Pdu_SvcType (fda_pdu) = "AR Establish Req" 625) && FAL_PDU_Valid(fda_pdu) = False 626) => 627) FAL-PDU_req { 628) smpm_service_name = "SMPM_Send", 629) arep_id = GetArepId (), 630) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 631) fda_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 632) error_class = "service", 633) error_code = appropriate error code for detected error) 634) }	635) CLOSED
636) R3	637) CLOSED	638) FAL-PDU_ind 639) && EndpointType = "SERVER" 640) && FAL_Pdu_SvcType (fal_pdu) = "AR Establish Req" 641) && ConfigurationArCheckOK (fal_pdu) = False 642) => 643) FAL-PDU_req { 644) smpm_service_name = "SMPM_Send", 645) arep_id = GetArepId (), 646) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 647) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 648) error_class = "access", 649) error_code = "config access already open") 650) }	651) CLOSED
652) R4	653) CLOSED	654) FAL-PDU_ind 655) && EndpointType = "SERVER" 656) && FAL_Pdu_SvcType (fal_pdu) = "AR Establish Req" 657) && ConfigurationArCheckOK(fal_pdu) = True 658) => 659) RemoteAddress = FAL_Pdu_RemoteAddress () 660) EST_ind* { 661) arep_id = GetArepId (), 662) user_data = GetUserData (fal_pdu) 663) } 664) StartInactivityCloseTimer() 665) 666) * Les négociations des paramètres relèvent de l'entité de serveur qui reçoit cette indication.	667) RESPON DING
668) R5	669) OPEN 670) RESPONDING 671) REQUESTING	672) FAL-PDU_ind 673) && FAL_Pdu_SvcType (fal_pdu) = "AR Establish Req" 674) => 675) FAL-PDU_req { 676) smpm_service_name = "SMPM_Send", 677) arep_id = GetArepId (), 678) remote_address = FAL_Pdu_RemoteAddress (), 679) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 680) error_class = "service", 681) error_code = "object state conflict") 682) }	683) CLOSED

#	État courant	Événement ou condition => action	État suivant
684) R6	685) REQUESTING	686) FAL-PDU_ind 687) && FAL_Pdu_SvcType (fal_pdu) = “AR Establish Err” 688) => 689) EST_cnf(-) { 690) arep_id = GetArepId (), 691) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 692) }	693) CLOSED
694) R7	695) REQUESTING	696) FAL-PDU_ind 697) && FAL_Pdu_SvcType (fal_pdu) = “AR Establish Rsp” 698) => 699) BufferSize = FAL_Pdu_BufferSize(fal_pdu) 700) InactivityCloseTime = GetInactivityCloseTime(fal_pdu) 701) EST_cnf(+) { 702) arep_id = GetArepId (), 703) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 704) } 705) StartInactivityCloseTimer()	706) OPEN
707) R8	708) RESPONDING 709) REQUESTING	710) FDA-PDU_ind 711) && FDA_Pdu_SvcType (fda_pdu) = “CS_ReqPDU” 712) => 713) FDA-PDU_req { 714) smpm_service_name = “SMPM_Send”, 715) arep_id = GetArepId (), 716) remote_address = FDA_Pdu_RemoteAddress (fda_pdu), 717) fda_pdu = BuildFDA-ErrPDU(request_pdu = fda_pdu, 718) error_class = “service”, 719) error_code = “object state conflict”) 720) } 721) StartInactivityCloseTimer()	722) CLOSED
723) R9	724) RESPONDING	725) FAL-PDU_ind 726) && FAL_Pdu_SvcType (fal_pdu) = “ANY FAL PDU” 727) =>	728) CLOSED
729) R10	730) OPEN	731) FAL-PDU_ind 732) && FAL_Pdu_SvcType (fal_pdu) = “UCS_ReqPDU” 733) && FAL_Pdu_GetVcrId() <> NULL 734) => 735) UCS_ind { 736) arep_id = GetArepId (), 737) vcr_id = FAL_Pdu_GetVcrId(), 738) service_type = FAL_Pdu_SvcType(fal_pdu), 739) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 740) } 741) StartInactivityCloseTimer()	742) OPEN
743) R11	744) OPEN	745) FAL-PDU_ind 746) && FAL_Pdu_SvcType (fal_pdu) = “UCS_ReqPDU” 747) && FAL_Pdu_GetVcrId() = NULL 748) => 749) FAL-PDU_req { 750) smpm_service_name = “SMPM_Send”, 751) arep_id = GetArepId (), 752) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 753) fal_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 754) error_class = “access”, 755) error_code = “unrecognized FDA Address”) 756) } 757) StartInactivityCloseTimer()	758) OPEN
759) R12	760) OPEN	761) FAL-PDU_ind 762) && FAL_Pdu_SvcType (fal_pdu) = “UCS_ReqPDU” 763) => 764) CS_ind { 765) arep_id = GetArepId (), 766) vcr_id = FAL_Pdu_GetVcrId(), 767) service_type = FAL_Pdu_SvcType(fal_pdu), 768) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 769) } 770) StartInactivityCloseTimer()	771) OPEN
772) R13	773) OPEN	774) FAL-PDU_ind 775) && FAL_Pdu_SvcType (fda_pdu) = “DTC_ReqPDU” 776) && MaxOutstandingReached() = True 777) => 778) FAL-PDU_req { 779) smpm_service_name = “SMPM_Send”,	787) OPEN

#	État courant	Événement ou condition => action	État suivant
		780) arep_id = GetArepId (), 781) remote_address = FAL_Pdu_RemoteAddress (fal_pdu), 782) fda_pdu = BuildFAL-ErrPDU(request_pdu = fal_pdu, 783) error_class = “resource”, 784) error_code = “max outstanding requests per session exceeded”) 785) } 786) StartInactivityCloseTimer()	
788) R14	789) OPEN	790) FAL-PDU_ind 791) && (FAL_Pdu_SvcType (fal_pdu) = “CS_RspPDU” 792) || FAL_Pdu_SvcType (fal_pdu) = “CS_ErrPDU) 793) => 794) CS_cnf { 795) arep_id = GetArepId (), 796) vcr_id = FAL_Pdu_GetVcrId(), 797) service_type = FAL_Pdu_SvcType(fal_pdu), 798) user_data = FAL_Pdu_ServiceSpecificParameters(fal_pdu) 799) } 800) StartInactivityCloseTimer()	801) OPEN
802) R15	803) OPEN REQUESTING 804) RESPONDING	805) ErrorToARPM 806) && ErrorType = “socket disconnect” 807) =>	808) CLOSED

4.8.2 ARPM

4.8.2.1 Généralités

L’ARPM est utilisée par les AR Publisher/Subscriber et Report Distribution.

4.8.2.2 Définitions des primitives

4.8.2.2.1 Primitives échangées entre ARPM et FSPM

Les Tableaux 136 et 137 représentent les primitives échangées entre ARPM et FSPM.

Tableau 136 – Primitives émises par FSPM vers ARPM

Nom de la primitive	Source	Paramètres associés	Fonctions
EST_req	FSPM	arep_id, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish request" de la FSPM vers l’ARPM.
Abort_req	FSPM	vcr_id, abort_detail, abort_identifier, reason_code	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Abort request" de la FSPM vers l’ARPM.
UCS_req	FSPM	vcr_id, service_type, user_data	809) C'est une primitive interne de FAL utilisée pour acheminer une primitive " Unconfirmed Send (CS) request" de la FSPM vers l’ARPM.

Tableau 137 – Primitives émises par ARPM vers FSPM

Nom de la primitive	Source	Paramètres associés	Fonctions
EST_cnf(+)	ARPM	arep_id, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Establish response(+)" du ARPM vers le FSPM.
Abort_ind	ARPM	vcr_id, abort_detail, abort_identifier, reason_code	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Abort" de la FSPM vers l'ARPM.
UCS_ind	ARPM	vcr_id, service_type, user_data	C'est une primitive interne de FAL utilisée pour acheminer une primitive "Unconfirmed Send (CS)" de la FSPM vers l'ARPM.

4.8.2.2.2 Paramètres des primitives de FSPM/ARPM

Les paramètres utilisés avec les primitives échangées entre FSPM et ARPM sont décrits dans le Tableau 138.

Tableau 138 – Paramètres utilisés avec les primitives échangées entre FSPM et ARPM

Nom du paramètre	Description
arep_id	Ce paramètre est utilisé pour l'identification non ambiguë d'une instance d'AREP ayant émis une primitive. Le moyen d'une telle identification n'est pas spécifié par la présente norme.
vcr_id	Ce paramètre est utilisé pour l'identification non ambiguë d'une instance de VCR ayant émis une primitive ou la destination de la primitive. Le moyen d'une telle identification n'est pas spécifié par la présente norme.
user_data	Ce paramètre achemine les données d'utilisateur d'AR (y compris les paramètres spécifiques au service).
abort_detail	Ce paramètre achemine la valeur utilisée pour le paramètre Abort_Detail.
abort Identifier	Ce paramètre achemine la valeur utilisée pour le paramètre Identifier.
reason_code	Ce paramètre achemine la valeur utilisée pour le paramètre Reason_Code.
remote_address	Ce paramètre achemine l'adresse distante de la couche sous-jacente.
service_type	Ce paramètre achemine le protocole Id et le service dans les limites du protocole.

4.8.2.3 Mapping de DLL de la classe MulticastAREP

4.8.2.3.1 Définition formelle

Le Paragraphe 4.8.2.3 décrit le mapping des classes AREP Publisher / Subscriber and Report Distribution vers le modèle Socket. Le paragraphe 4.8.2.3 définit les attributs du mapping Socket et leurs valeurs autorisées et utilisées avec les classes AREP Publisher / Subscriber and Report Distribution.

CLASS : **Multicast**
PARENT CLASS : Point d'extrémité AR
ATTRIBUTES:

1	(m)	KeyAttribute:	LocalAndRemoteAddresses
1,1	(m)	KeyAttribute:	LocalAddress
1,2	(m)	KeyAttribute:	RemoteAddress
2	(m)	Attribut:	TransferType (CONN, CNLS)
4	(c)	Contrainte:	Role = SUBSCRIBER | REPORT SINK
4,1	(m)	Attribut:	GuardBand
5	(c)	Contrainte:	Role = PUBLISHER | REPORT SOURCE
5,1	(m)	Attribut:	TransmitDelayTime
6	(m)	Attribut:	BufferSize
7	(m)	Attribut:	HdrOptions
8	(c)	Contrainte:	Role = SUBSCRIBER | REPORT SINK
8,1	(m)	Attribut:	MaxApduLength

4.8.2.3.2 Attributs

LocalAndRemoteAddresses

Cet attribut clé identifie l'AREP. Il est composé des adresses distantes et locales de la couche sous-jacente pour l'AREP.

TransferType

Cet attribut spécifie si les services orientés connexion (CONN) ou sans connexion (CNLS) des couches sous-jacentes sont utilisés par l'AREP.

GuardBand

Cet attribut conditionnel spécifie la limite inférieure et supérieure pour le champ protégé du repassage par zéro («rollover»). Il est présent si ROLE est SUBSCRIBER ou REPORT SINK.

TransmitDelayTime

Cet attribut conditionnel spécifie la quantité de temps accumulée par un point d'extrémité de la session pour envoyer des messages dans son tampon. Il est présent, si ROLE est représenté par PUBLISHER ou REPORT SOURCE. La période de temps commence lorsque le premier message est placé dans le tampon émetteur soit après l'initialisation du tampon, soit après avoir été vidé par la transmission précédente.

Si le point d'extrémité n'a pas de messages à envoyer (le tampon est vide), mais il reçoit un message à envoyer, il le charge dans le tampon et attend des messages supplémentaires à enchaîner dans le tampon. Si le tampon se remplit avant que Transmit Delay Time soit atteint, FAL AE de Type 5 arrête le chronomètre et envoie le tampon. Une fois l'intervalle de temps expiré, alors que le tampon n'est pas encore plein, il envoie le contenu du tampon.

BufferSize

Cet attribut spécifie la taille en octets du tampon d'envoi du point d'extrémité de la session. Le tampon possède un nombre entier de messages. Si c'est un message reçu pour le transfert qui ferait déborder le tampon, le tampon est envoyé en premier, et le message s'ajoute après en étant le premier message dans le tampon.

HdrOptions

Cet attribut spécifie les champs de fin de trame des messages présents dans les messages envoyés et reçus sur la session.

MaxApduLength

Cet attribut conditionnel spécifie la longueur maximale autorisée en octets des APDU envoyées sur l'AR. Il est présent si ROLE est SUBSCRIBER ou REPORT SINK.

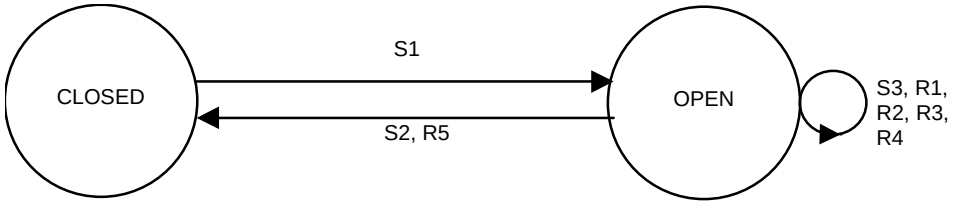
4.8.2.4 Diagramme d'états de MulticastARPM

4.8.2.4.1 États de MulticastARPM

Les états définis et leurs descriptions de l'ARPM Client / Serveur sont représentés dans le Tableau 139 et la Figure 3. Pour CONN, l'état fermé n'est jamais abandonné jusqu'à ce que la connexion soit établie. La façon dont cette connexion est établie dépend de la mise en œuvre.

Tableau 139 – États d'ARPM éditer / abonné

CLOSED	L'AREP est défini, mais incapable d'envoyer ou de recevoir des FAL-PDU arbitraires. Il peut envoyer ou recevoir des FAL-PDU du service Establish en étant dans cet état.
OPEN	L'AREP est défini et est capable d'envoyer ou de recevoir des FAL-PDU.



Légende

Anglais	Français
CLOSED	Fermé
OPEN	ouvert

Figure 3 – Diagramme de transition d'états d'ARPM éditeur / abonné

4.8.2.4.2 Table d'états d'ARPM Multicast

Les Tableaux 140 et 141 définissent le diagramme d'états d'ARPM Multicast.

Tableau 140 – Table d'états de MulticastARPM – transitions d'expéditeur

#	État courant	Événement ou condition => action	État suivant
810) S1	811) CLOSED	812) EST_req 813) => 814) EST_cnf(+){ 815) arep_id := GetArepId (), 816) user_data := "null" 817) }	818) OPEN
819) S2	820) OPEN 821)	822) Abort_req 823) =>	824)
825) S3	826) OPEN	827) UCS_req 828) && Role = "PUBLISHER" || "REPORT SOURCE" 829) => 830) FAL-PDU_req { 831) dmpm_service_name := "SMPM_Send", 832) arep_id := GetArepId (), remote_address := RemoteAddress*, 833) fal_pdu := BuildFAL-ReqRspPDU (fal_service_type = service_type, 834) fal_data := user_data) 835) }	836) OPEN

Tableau 141 – Table d'états de MulticastARPM – transitions de destinataire

#	État courant	Événement ou condition => action	État suivant
837) R1	838) OPEN	839) FAL-PDU_ind 840) && FAL_Pdu_Valid(fal_pdu) = False 841) => 842) Report to local management	843) OPEN
844) R2	845) OPEN	846) FAL-PDU_ind 847) && Endpoint Type = "SUBSCRIBER" || "REPORT SINK" 848) && FAL_Pdu_Confirmed(fal_pdu) = True 849) => 850) Report to local management	851) OPEN
852) R3	853) OPEN	854) FAL-PDU_ind 855) && Endpoint Type = "SUBSCRIBER" 856) && FAL_Pdu_GetVcrId() <> NULL 857) => 858) UCS_ind* { 859) arep_id = GetArepId (), 860) vcr_id = GetVcrId (), 861) service_type = GetServiceType(fal_pdu), 862) user_data = GetUserData(fal_pdu) 863) } 864) 865)	866) OPEN
867) R4	868) OPEN	869) FAL-PDU_ind 870) && Endpoint Type = "REPORT SINK" 871) => 872) UCS_ind* { 873) arep_id = GetArepId (), 874) vcr_id = GetVcrId (), 875) service_type = GetServiceType(fal_pdu), 876) user_data = GetUserData(fal_pdu) 877) } 878) 879) * Livré à tous les VCR associés à cet AREP	880) OPEN
881) R5	882) OPEN	883) ErrorToARPM 884) && ErrorType = "socket disconnect" 885) => 886) Abort_ind* { 887) arep_id = GetArepId (), 888) locally_generated = "True", 889) identifier = "FAL", 890) reason_code = "Socket disconnected" 891) } 892) 893) * Livré à tous les VCR associés à cet AREP	894) CLOSED

4.8.3 Fonctions utilisées par les ARPM

Les Tableaux 142 à 158 définissent les fonctions utilisées par les ARPM.

4.8.3.1 BuildFAL-ErrPDU()

Tableau 142 – BuildFAL-ErrPDU()

Fonction	
Renvoie fal_pdu qui est une réponse d'erreur avec les erreurs spécifiées. Le paramètre d'entrée request_pdu fournit des informations nécessaires pour construire la pdu de réponse d'erreur appropriée.	
Paramètres d'entrée	Paramètres de sortie
request_pdu error_class error_code	fal_pdu

4.8.3.2 BuildFAL-ReqRspPDU()

Tableau 143 – BuildFAL-ReqRspPDU()

Fonction	
Construit une FAL PDU à partir des paramètres donnés comme variables d'entrée. Cela inclut : Construction de l'en-tête pdu de FAL. Ajout de fal_data (données d'utilisateur d'AR) Production de la fin de trame de FAL_PDU.	
Paramètres d'entrée	Paramètres de sortie
service_type fal_data	fal_pdu

4.8.3.3 GetArepId()

Tableau 144 – GetArepId()

Fonction	
Renvoie une valeur qui permet l'identification non ambiguë de l'AREP en cours.	
Paramètres d'entrée	Paramètres de sortie
None	AREP Identifier

4.8.3.4 ConfigurationArCheckOK()

Tableau 145 – ConfigurationArCheckOK()

Fonction	
Renvoie "True" si : fal_pdu de demande n'est pas une demande de configuration. fal_pdu de demande est une demande de configuration et il n'existe pas d'autres AR de configuration établies.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	True | False

4.8.3.5 FAL_Pdu_BufferSize()

Tableau 146 – FAL_Pdu_BufferSize()

Fonction	
Cette fonction renvoie la taille du tampon à partir d'AR Establish fal_pdu.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	BufferSize

4.8.3.6 FAL_Pdu_Confirmed()

Tableau 147 – FAL_Pdu_Confirmed()

Fonction	
Cette fonction renvoie vrai si le fal_pdu est confirmé.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	True | False

4.8.3.7 FAL_Pdu_DuplicateMsg ()

Tableau 148 – FAL_Pdu_DuplicateMsg ()

Fonction	
Cette fonction renvoie "True" si fal_pdu reçu est défini comme une copie basée sur son numéro de message et le paramètre vcr_id. Sinon, elle renvoie "False".	
Paramètres d'entrée	Paramètres de sortie
vcr_id, fal_pdu	True | False

4.8.3.8 FAL_Pdu_GetVcrId()

Tableau 149 – FAL_Pdu_GetVcrId()

Fonction	
Renvoie une valeur qui identifie le point d'extrémité de VCR associé à une AR, en utilisant l'adresse de FDA dans l'en-tête de fal_pdu. Peut renvoyer une liste si l'AREP est un abonné. Renvoie "NULL", si FDA Address n'identifie pas le point d'extrémité VCR.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	VCR Identifier

4.8.3.9 FAL_Pdu_InactivityCloseTime()

Tableau 150 – FAL_Pdu_InactivityCloseTime()

Fonction	
Cette fonction renvoie InactivityCloseTime à partir d'AR Establish fal_pdu.	
Paramètres d'entrée	Paramètres de sortie
AR Establish fal_pdu	InactivityCloseTime

4.8.3.10 FAL_Pdu_TransmitDelayTime()

Tableau 151 – FAL_Pdu_TransmitDelayTime()

Fonction	
Cette fonction renvoie TransmitDelayTime à partir d'AR Establish fal_pdu.	
Paramètres d'entrée	Paramètres de sortie
AR Establish fal_pdu	TransmitDelayTime

4.8.3.11 FAL_Pdu_SvcType()

Tableau 152 – FAL_Pdu_SvcType()

Fonction	
Cette fonction décode FAL-PDU, qui est acheminée dans le paramètre fall_pdu, et récupère le type de service de FAL-PDU.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	service type

4.8.3.12 FAL_Pdu_RemoteAddress()

Tableau 153 – FAL_Pdu_RemoteAddress()

Fonction	
Renvoie l'adresse de l'expéditeur de fal_pdu reçu.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	address of sender

4.8.3.13 FAL_Pdu_TrailerFields()

Tableau 154 – FAL_Pdu_TrailerFields()

Fonction	
La fonction renvoie les champs de fin de trame de fal_pdu.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	Les champs de fin de trame présents dans FAL_PDU.

4.8.3.14 FAL_Pdu_ServiceSpecificParameters()

Tableau 155 – FAL_Pdu_ServiceSpecificParameters()

Fonction	
Cette fonction renvoie les paramètres spécifiques au service, y compris les données d'utilisateur, à partir d'un fal_pdu.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	Paramètres spécifiques au service

4.8.3.15 FAL_Pdu_Valid()

Tableau 156 – FAL_Pdu_Valid()

Fonction	
Cette fonction renvoie "False" si l'ARPM peut déterminer que : 1. l'en-tête ou la fin de trame de fal_pdu contient des valeurs non valides ou incohérentes, ou 2. l'en-tête ou la fin de trame de fal_pdu contient des options qui ne correspondent pas à celles négociées pour l'AR. 3. fal_pdu ne peut pas être décodé ou identifié correctement. Les paramètres spécifiques au service ne sont pas examinés par cette fonction.	
Paramètres d'entrée	Paramètres de sortie
fal_pdu	True | False

4.8.3.16 MaxOutstandingReached()

Tableau 157 – MaxOutstandingReached()

Fonction	
La fonction vérifie le compteur local pour le numéro actuel des demandes en cours pour déterminer s'il a atteint la limite définie par l'attribut de la session MaxOutstanding. Elle renvoie "True", si la limite a été atteinte. Sinon, elle incrémente le compteur local pour le numéro actuel des demandes en cours et renvoie "False".	
Paramètres d'entrée	Paramètres de sortie
None	True | False

4.8.3.17 StartInactivityCloseTimer()

Tableau 158 – StartInactivityCloseTimer()

Fonction	
La fonction (re)lance un chronomètre pour une période de temps définie par Inactivity Close Time.	
Paramètres d'entrée	Paramètres de sortie
None	None

4.9 Machine protocolaire de mapping de DLL (DMPM)

4.9.1 Généralités

Le mapping de DLL est représenté par le modèle Socket.

Le modèle Socket définit une interface de service abstraite qui autorise l'existence de la couche sous-jacente sur la couche liaison de données, la couche réseau, ou la couche transport. Deux types de services sont pris en charge par le modèle Socket: orienté connexion (CONN et sans connexion (CNLS).

4.9.2 Définitions des primitives

4.9.2.1 Primitives échangées entre DMPM et ARPM

Les Tableaux 159 et 160 définissent les primitives échangées entre DMPM et ARPM.

Tableau 159 – Primitives émises par ARPM vers DMPM

Nom de la primitive	Source	Paramètres associés	Fonctions
FAL-PDU_req	ARPM	smpm_service_name, remote_address, fal_pdu	Cette primitive est utilisée pour demander le DMPM afin de transférer une FAL-PDU. La liste de smpm_service_names est : SMPM_Send

Tableau 160 – Primitives émises par DMPM vers ARPM

Nom de la primitive	Source	Paramètres associés	Fonctions
FAL-PDU_ind	DMPM	smpm_service_name, remote_address, fal_pdu	Cette primitive est utilisée pour faire passer une FAL-PDU reçue comme une unité de données de service du modèle Socket vers l'ARPM désignée. Il transporte également certains paramètres du modèle Socket qui sont référencés dans l'ARPM. La liste de smpm_service_names est : SMPM_Receive
ErrorToARPM	DMPM	originator, reason	Cette primitive est utilisée pour acheminer des erreurs de communication sélectionnées par le modèle Socket vers un ARPM désigné.

4.9.2.2 Paramètres des primitives d'ARPM/DMPM

Les paramètres utilisés avec les primitives échangées entre ARPM et DMPM sont décrits dans le Tableau 161.

Tableau 161 – Paramètres utilisés avec les primitives échangées entre ARPM et DMPM

Nom du paramètre	Description
fal_pdu	Ce paramètre achemine la valeur du paramètre socket_user_data.
smpm_service_name	Ce paramètre achemine un nom du service du modèle Socket.
remote_address	Ce paramètre achemine la valeur de l'adresse distante de la couche sous-jacente.
Originator	Ce paramètre identifie l'entité qui a détecté l'erreur, ce qui est rapporté à l'aide de la primitive ErrorToARPM.
Reason	Ce paramètre identifie la cause qui a détecté l'erreur, ce qui est rapporté à l'aide de la primitive ErrorToARPM.

4.9.2.3 Primitives échangées entre le modèle socket et DMPM

Les définitions sont représentées dans le Tableau 162.

NOTE 1 Les primitives suivantes et leurs paramètres sont échangés entre le modèle Socket et la DMPM.

NOTE 2 L'instance de DMPM utilise soit les services CNLS, soit les services CONN.

NOTE 3 Chaque ARPM est supposée être liée à l'adresse locale et n'est jamais un paramètre d'une primitive.

Tableau 162 – Primitives échangées entre le modèle socket et DMPM

Nom de la primitive	Source	Paramètres associés
socket_send.ind	Socket model	socket_user_data
socket_sendto.ind	Socket model	socket_remote_address, socket_user_data
socket_sendto.req	DMPM	socket_remote_address, socket_user_data
socket_send.req	DMPM	socket_user_data

4.9.2.4 Paramètres des primitives du modèle DMPM/Socket

Les définitions sont représentées dans le Tableau 163.

Tableau 163 – Paramètres des primitives du modèle DMPM/Socket

Nom du paramètre	Description
socket_remote_address	Ce paramètre achemine la valeur de l'adresse distante du modèle Socket.
socket_user_data	Le message complet que le modèle socket doit envoyer à / recevoir d'une destination spécifiée.

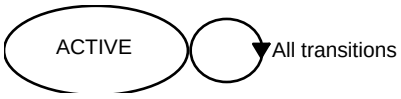
4.9.3 Diagramme d'états de DMPM

4.9.3.1 États de DMPM

L'état défini et la description de DMPM sont représentés dans le Tableau 164 et la Figure 4.

Tableau 164 – Descriptions des états de DMPM

Nom de l'état	Description
ACTIVE	Le DMPM dans un état ACTIVE est prêt à transmettre ou recevoir des primitives vers ou depuis le modèle Socket et l'ARPM.



Légende

Anglais	Français
ACTIVE	Actif
All transitions	Toutes transitions

Figure 4 – Diagramme de transition d'états de DMPM

4.9.3.2 Table d'états de DMPM

Les Tableaux 165 et 166 définissent le diagramme d'états de DMPM.

NOTE 1 Bien que chaque primitive contienne tous les paramètres disponibles, seuls les paramètres applicables à des ARPM particulières sont pertinents.

NOTE 2 Des paramètres qui commencent par une lettre majuscule, par exemple, "TransmitDelayTime", se réfèrent aux paramètres définis dans la liste d'attributs pour chaque ARPM. C'est pourquoi ils ne sont pas acheminés par les primitives de services définies ici.

Tableau 165 – Table d'états de DMPM – transitions d'expéditeur

#	État courant	Événement ou condition => action	État suivant
895) S1	896) ACTIVE	897) FDA-PDU_req 898) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "FIRST MESSAGE" 899) => 900) LoadBuffer(arep_id, fal_pdu) 901) StartTransmitDelayTimer(arep_id)	902) ACTIVE
903) S2	904) ACTIVE	905) FDA-PDU_req 906) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "LESS THAN" 907) => 908) LoadBuffer(arep_id, fal_pdu)	909) ACTIVE
910) S3	911) ACTIVE	912) FDA-PDU_req 913) && ConnectionOriented() = True 914) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "EQUAL" 915) => 916) socket_send.req { 917) user_data = GetBufferedData(arep_id) 918) }	919) ACTIVE
920) S4	921) ACTIVE	922) FDA-PDU_req 923) && ConnectionOriented() = FALSE 924) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "EQUAL" 925) => 926) socket_sendto.req { 927) remote_address = remote_address, 928) user_data = GetBufferedData(arep_id) 929) }	930) ACTIVE
931) S5	932) ACTIVE	933) FDA-PDU_req 934) && ConnectionOriented() = True 935) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "GREATER THAN" 936) => 937) socket_send.req { 938) user_data = fal_pdu 939) } 940) LoadBuffer(arep_id,fal_pdu) 941) StartTransmitDelayTimer(arep_id)	942) ACTIVE
943) S6	944) ACTIVE	945) FDA-PDU_req 946) &&ConnectionOriented()=FALSE 947) && RemainingBufferSizeCheck (arep_id, fal_pdu) = "GREATER THAN" 948) => 949) socket_sendto.req { 950) remote_address = remote_address, 951) user_data = GetBufferedData(arep_id) 952) } 953) LoadBuffer(arep_id, fal_pdu) 954) StartTransmitDelayTimer(arep_id)	955) ACTIVE
956) S7	957) ACTIVE	958) Transmit delay timer expire 959) && ConnectionOriented = False 960) => 961) socket_sendto.req { 962) remote_address = GetRemoteAddress(arep_id), 963) user_data = GetBufferedData(arep_id) 964) }	965) ACTIVE
966) S8	967) ACTIVE	968) Transmit delay timer expire 969) && ConnectionOriented() = True 970) => 971) socket_send.req { 972) user_data = GetBufferedData(arep_id) 973) }	974) ACTIVE

Tableau 166 – Table d'états de DMPM – transitions de destinataire

#	État courant	Événement ou condition => action	État suivant
975) R1	976) ACTIVE	977) socket_sendto.ind 978) => 979) FAL-PDU_ind { 980) smpm_service_name := "SMPM_Receive", 981) remote_address := socket_remote_address, 982) fal_pdu := socket_user_data 983) }	984) ACTIVE
985) R2	986) ACTIVE	987) socket_send.ind 988) => 989) FAL-PDU_ind { 990) smpm_service_name := "SMPM_Receive", remote_address := GetConnectionId(arep_id) , 991) fal_pdu := socket_user_data 992) }	993) ACTIVE
994) R3	995) ACTIVE	996) "Connection breaks" 997) => 998) ErrorToArpm { 999) originator= local_socket, 1000) reason = "Socket disconnected" 1001) }	1002) ACTIVE

4.9.3.3 Fonctions utilisées par DMPM

Les Tableaux 167 à 172 définissent les fonctions utilisées par la DMPM.

4.9.3.3.1 Fonction ConnectionOriented

Tableau 167 – ConnectionOriented

Nom	ConnectionOriented	Utilisé dans	ARPM
Entrée		Sortie	
		True | False	
Fonction			
1003) Cette fonction renvoie "True", si l'instance de SMPM est orientée connexion (CONN) et "False", si l'instance de SMPM est sans connexion (CNLS).			

4.9.3.3.2 Fonction GetBufferedData

Tableau 168 – GetBufferedData

Fonction	
Cette fonction renvoie le contenu du tampon d'envoi d'AREP et marque le tampon comme vide.	
Paramètres d'entrée	Paramètres de sortie
ArepId	buffered_data

4.9.3.3.3 Fonction GetConnectionId

Tableau 169 – GetConnectionId

Fonction	
Cette fonction renvoie la connexion id d'un socket orienté connexion.	
Paramètres d'entrée	Paramètres de sortie
ArepId	Remote_address

4.9.3.3.4 Foncion LoadBuffer

Tableau 170 – LoadBuffer

Fonction	
Cette fonction enchaîne user_data dans le tampon d'AREP.	
Paramètres d'entrée	Paramètres de sortie
ArepId, user_data	None

4.9.3.3.5 Fonction RemainingBufferSizeCheck

Tableau 171 – RemainingBufferSizeCheck

Fonction	
Cette fonction compare la taille de user_data avec la taille de l'espace qui reste dans le tampon d'envoi d'AREP. Elle renvoie : "FIRST MESSAGE" le tampon est actuellement vide et la taille de user_data est inférieure à la taille du tampon. "EQUAL" la taille de user_data est égale à la taille de l'espace qui reste dans le tampon. "LESS THAN" la taille de user_data est inférieure à la taille de l'espace qui reste dans le tampon. "GREATER THAN" la taille de user_data est supérieure à la taille de l'espace qui reste dans le tampon.	
Paramètres d'entrée	Paramètres de sortie
ArepId, user_data	buffer_status

4.9.3.3.6 StartTransmitDelayTimer

Tableau 172 – StartTransmitDelayTimer

Fonction	
Cette fonction commence ou recommence Transmit Delay Timer de l'AREP identifié par le paramètre ArepId.	
Paramètres d'entrée	Paramètres de sortie
ArepId	None

Bibliographie

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 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/IEC 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de la notation de syntaxe abstraite numéro 1 (ASN.1)*[1]

_____

[1] Retirée.

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