IEC 61158-6-4

Edition 3.0 2019-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

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

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-4: Spécification du protocole de la couche application – Eléments de type 4

IEC 61158-6-4:2019-06(en-fr)

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

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

IEC 61158-6-4

Edition 3.0 2019-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

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

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-4: Spécification du protocole de la couche application – Eléments de type 4

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-9702-5

CONTENTS

FOREWORD 5
INTRODUCTION 7
1 Scope 8
1.1 General 8
1.2 Specifications 8
1.3 Conformance 9
2 Normative references 9
3 Terms, definitions, symbols, abbreviations and conventions 9
3.1 Referenced terms and definitions 10
3.1.1 ISO/IEC 7498-1 terms 10
3.1.2 ISO/IEC 8822 terms 10
3.1.3 ISO/IEC 9545 terms 10
3.1.4 ISO/IEC 8824-1 terms 10
3.1.5 Fieldbus data-link layer terms 10
3.2 Abbreviations and symbols 11
3.3 Conventions 11
3.3.1 General concept 11
3.3.2 Conventions for state machines for Type 4 12
4 FAL syntax description 13
4.1 FAL-AR PDU abstract syntax 13
4.1.1 General 13
4.1.2 Abstract syntax of APDU header 13
4.1.3 Abstract syntax of APDU body 14
4.2 Data types 16
5 Transfer syntaxes 16
5.1 APDU encoding 16
5.1.1 APDU Header encoding 16
5.1.2 APDU body encoding 18
5.2 Variable object encoding and packing 20
5.2.1 Encoding of simple variables 20
5.2.2 Encoding of constructed variables 21
5.2.3 Alignment 22
5.2.4 Variable object attributes 23
5.3 Error codes 24
6 FAL protocol state machines 25
7 AP-context state machine 26
8 FAL service protocol machine (FSPM) 27
8.1 Primitives exchanged between FAL User and FSPM 27
8.2 FSPM states 27
8.2.1 General 27
8.2.2 FSPM proxy object states 27
8.2.3 FSPM real object state machine description 32
9 Application relationship protocol machine (ARPM) 34
9.1 Primitives exchanged between ARPM and FSPM 34
9.2 ARPM States 34
9.2.1 General 34

9.2.2 Sender state transitions 34
9.2.3 Receiver state transitions 35
10 DLL mapping protocol machine (DMPM) 36
10.1 Data-link Layer service selection 36
10.1.1 General 36
10.1.2 DL-UNITDATA request 36
10.1.3 DL-UNITDATA indication 36
10.1.4 DL-UNITDATA response 36
10.1.5 DLM-Set primitive and parameters 36
10.1.6 DLM-Get primitive and parameters 36
10.2 Primitives exchanged between ARPM and DLPM 36
10.3 Primitives exchanged between DLPM and data-link layer 37
10.4 DLPM states 37
10.4.1 States 37
10.4.2 Sender state transitions 38
10.4.3 Receiver state transitions 39
11 Protocol options 39
Bibliography 40

Figure 1 – State transition diagram 12
Figure 2 – APDU header structure 16
Figure 3 – Subfields of ControlStatus for Request 17
Figure 4 – Subfields of ControlStatus for Response with error 17
Figure 5 – Subfields of ControlStatus for Response with no error 18
Figure 6 – DataFieldFormat encoding 18
Figure 7 – Structure of request APDU body 19
Figure 8 – Structure of response APDU body 19
Figure 9 – Variable identifier 19
Figure 10 – Code subfield of variable identifier 19
Figure 11 – Sequence of data in the APDU body subfield 21
Figure 12 – MSG consists of APDU header and APDU body 22
Figure 13 – Summary of FAL architecture 26
Figure 14 – FSPM proxy object state machine 28
Figure 15 – FSPM real object state machine 33
Figure 16 – ARPM state machine 34
Figure 17 – DLPM state machine 37

Table 1 – State machine description elements 12
Table 2 – APDU header 13
Table 3 – APDU body 15
Table 4 – Transfer syntax for Array 23
Table 5 – Transfer syntax for Structure 23
Table 6 – Common variable object attributes 23
Table 7 – Variable type identifiers 24
Table 8 – FIFO variable object attributes 24

Table 9 – Error codes 25
Table 10 – Primitives exchanged between FAL-User and FSPM 27
Table 11 – REQUEST.req FSPM constraints 28
Table 12 – REQUEST.req FSPM actions 29
Table 13 – RESPONSE.cnf FSPM constraints 30
Table 14 – RESPONSE.cnf FSPM actions 31
Table 15 – AR Send.ind proxy FSPM constraints 32
Table 16 – AR Send.ind proxy FSPM actions 32
Table 17 – AR Send.ind real FSPM constraints 33
Table 18 – AR Send.ind real FSPM Actions 33
Table 19 – Primitives issued by FSPM to ARPM 34
Table 20 – Primitives issued by ARPM to FSPM 34
Table 21 – Primitives issued by ARPM to ARPM 34
Table 22 – AR Send.req ARPM constraints 35
Table 23 – AR Send.req ARPM actions 35
Table 24 – AR Acknowledge.req ARPM constraints 35
Table 25 – AR Acknowledge.req ARPM actions 35
Table 26 – AR Send.ind ARPM constraints 36
Table 27 – AR Send.req ARPM actions 36
Table 28 – Primitives issued by ARPM to DLPM 37
Table 29 – Primitives issued by DLPM to ARPM 37
Table 30 – Primitives issued by DLPM to data-link layer 37
Table 31 – Primitives issued by data-link layer to DLPM 37
Table 32 – AR Send.req DLPM constraints 38
Table 33 – AR Send.req DLPM actions 38
Table 34 – AR Acknowledge.req DLPM constraints 38
Table 35 – AR Acknowledge.req DLPM actions 39
Table 36 – DL-UNITDATA.ind DLPM constraints 39
Table 37 – DL-UNITDATA.ind DLPM actions 39

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-4: Application layer protocol specification – Type 4 elements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC 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 61158-6-4 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This third edition cancels and replaces the second edition published in 2014. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) additional user parameters to services;

b) additional services to support distributed objects;

c) additional secure services.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
65C/948/FDIS	65C/956/RVD

Full information on the voting for the approval of this International 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 document 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 document 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 document is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this document 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 6-4: Application layer protocol specification – Type 4 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 part of IEC 61158 provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 4 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This International Standard specifies interactions between remote applications and defines the externally visible behavior provided by the Type 4 fieldbus application layer in terms of

a) the formal abstract syntax defining the application layer protocol data units conveyed between communicating application entities;
b) the transfer syntax defining encoding rules that are applied to the application layer protocol data units;
c) the application context state machine defining the application service behavior visible between communicating application entities;
d) the application relationship state machines defining the communication behavior visible between communicating application entities.

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

1) define the wire-representation of the service primitives defined in IEC 61158-5-4, and
2) define the externally visible behavior associated with their transfer.

This document specifies the protocol of the Type 4 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this document is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-4.

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 document do not specify individual implementations or products, nor do they 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 are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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 61158-3-4:2019, *Industrial communication networks – Fieldbus specifications – Part 3-4: Data-link layer service definition – Type 4 elements*

IEC 61158-5-4:2019, *Industrial communication networks – Fieldbus specifications – Part 5-4: Application layer service definition – Type 4 elements*

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

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

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

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

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

ISO/IEC 9797-1, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*

3 Terms, definitions, symbols, abbreviations and conventions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 7498-1 apply:

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

3.1.2 ISO/IEC 8822 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8822 apply:

a) abstract syntax
b) presentation context

3.1.3 ISO/IEC 9545 terms

For the purposes of this document, the following terms as defined in ISO/IEC 9545 apply:

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

3.1.4 ISO/IEC 8824-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824-1 apply:

a) object identifier
b) type

3.1.5 Fieldbus data-link layer terms

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

a) DL-Time
b) DL-Scheduling-policy
c) DLCEP
d) DLC
e) DL-connection-oriented mode

f) DLPDU
g) DLSDU
h) DLSAP
i) network address
j) node address
k) node

3.2 Abbreviations and symbols

AE	Application Entity
AL	Application Layer
ALE	Application Layer Entity
APDU	Application Protocol Data Unit
AR	Application Relationship
AREP	Application Relationship End Point
ASE	Application Service Element
Cnf	Confirmation
DL-	(as a prefix) Data-link-
DLCEP	Data-link Connection End Point
DLL	Data-link Layer
DLE	Data-link Entity
DLM	Data-link-management
DLS	Data-link Service
DLSAP	Data-link Service Access Point
DLSDU	DL-service-data-unit
FME	FAL Management Entity
Ind	Indication
IP	Internet Protocol
PDU	Protocol Data Unit
Req	Request
Rsp	Response
SME	System Management Entity
.cnf	Confirm Primitive
.ind	Indication Primitive
.req	Request Primitive
.rsp	Response Primitive

3.3 Conventions

3.3.1 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 61158-5-4. The protocol specification for each of the ASEs is defined in this document.

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 61158-5-4. The service specification defines the services that are provided by the ASE.

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

3.3.2 Conventions for state machines for Type 4

A state machine describes the state sequence of an entity and can be represented by a state transition diagram and/or a state table.

In a state transition diagram (see Figure 1), the transition between two states represented by circles is illustrated by an arrow beside which the transition events or conditions are presented.

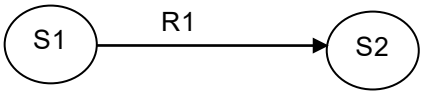


Figure 1 – State transition diagram

Table 1 – State machine description elements

#	Current state	Events or conditions that trigger this state transaction => 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 transition table (Table 1) 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
means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'
"xxx" Indicates fixed value.

Example:

Identifier:= "abc"
means value "abc" is assigned to a parameter named 'Identifier.'

= 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"

Service.req represents a Request Primitive; Service.req{} indicates that a request primitive is sent;

Service.ind represents an Indication Primitive; Service.ind{} indicates that an Indication Primitive is received;

Service.rsp represents a Response Primitive; Service.rsp{} indicates that a Response Primitive is sent;

Service.cnf represents a Confirm Primitive; Service.cnf{} indicates that a Confirm Primitive is received.

4 FAL syntax description

4.1 FAL-AR PDU abstract syntax

4.1.1 General

The information stored in an APDU depends on whether the APDU holds a request or a response. The role of the state machine that encodes the APDU (the FSPM) determines how the APDU is encoded.

APDUs always consist of an APDU header and an APDU body. In response APDUs the APDU body may be empty.

4.1.2 Abstract syntax of APDU header

Table 2 defines the contents of the APDU header.

Table 2 – APDU header

Field name	Subfield name	Possible values	Constraint (present if)	Comment
ControlStatus	Instruction	Errorcode Method Store Load And Or Test-And-Set Segmented Load Segmented Store		
ControlStatus	Errorcode	Described in Figure 3 to Figure 5	ControlStatus.Instruction = Errorcode	
ControlStatus	Addressing method	Variable Object Flat	ControlStatus.Instruction <> Errorcode	
ControlStatus	Addressing method	Variable Object Flat	ControlStatus.Instruction =Method	Variable object means 2 octet MethodID Flat means 4 octet MethodID

Field name	Subfield name	Possible values	Constraint (present if)	Comment
ControlStatus	SourceID	NoSourceIDInData SourceIDInData		Used by the requesting application to indicate to the responding application that the performed instruction can be related to a specific SourceID
ControlStatus	SecureDataExchange	NoSecureData SecureData	APDU is a request APDU	Read type instruction: Used by the requesting application to ensure an authenticated response. Write type instruction: Used by the requesting application to ensure an authenticated write type instruction in the responder.
ControlStatus	ActualDataError	NoActualError ActualError	ControlStatus.Instruction <> Errorcode	Used by the responding user application to indicate, that an actual error may affect the accessed Variable Object
ControlStatus	SystemResult	NoSystemResult SystemResult	ControlStatus.Instruction = Method	Used by the responding user application to indicate that additional 2 octet data are inserted in the APDU data before the result data for the accessed Variable identifier Hence, DataLength is 2 higher than the method result length.
ControlStatus	HistoricalDataError	NoHistoricalError HistoricalError	ControlStatus.Instruction <> Errorcode	Used by the responding user application to indicate, that an error may have affected the accessed Variable Object
DataFieldFormat	Offset/Attribute	No Offset/Attribute Offset/Attribute		Indicates, whether the APDU Body holds an Offset/Attribute field
DataFieldFormat	Variable Identifier Format	Simple Complex	APDU is a request APDU	Indicates the format of the Variable Identifier in a request APDU
DataFieldFormat	Offset/Attribute size	Integer16 Integer32	APDU is a response APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute	Indicates the size of the Offset/Attribute field of the APDU Body
DataLength		min. 2		Indicates the total length of the APDU Body. MaxDataSize indicates the max length of the data part of the APDU Body.

4.1.3 Abstract syntax of APDU body

The APDU header indicates the interpretation of the contents of the APDU body.

Table 3 defines the contents of the APDU body.

Table 3 – APDU body

Field name	Subfield name	Possible values	Constraint (present if)	Comment
VariableIdentifier	Code.Bitaddressing	No BitAddressing BitAddressing	APDU is a request APDU AND APDU Header indicates Complex VariableIdentifier	If this field indicates BitAddressing, the VariableIdentifier also holds a Bit-no
VariableIdentifier	Code.Bit-no	0 to 7	APDU is a request APDU AND APDU Header indicates Complex VariableIdentifier AND VariableIdentifier indicates BitAddressing	Bit-no selects a bit within one octet. Bit-no = 0 selects bit 1 etc. The octet is selected by Offset/Attribute.
VariableIdentifier	Code.Offset/Attribute size	Integer16 Integer32	APDU is a request APDU AND DataFieldFormat.Variable Identifier Format = Complex AND DataFieldFormat.Offset/Attribute = Offset/Attribute	
VariableIdentifier	ID	-32 768 to +32 767	APDU is a request APDU AND DataFieldFormat.Variable Identifier Format = Simple	
VariableIdentifier	ID	-8 388 608 to +8 388 607	APDU is a request APDU AND DataFieldFormat.Variable Identifier Format = Complex	
Offset/Attribute		-32 768 +32 767	APDU is a request APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND VariableIdentifier.Code.Offset/Attribute size = Integer16	Negative values select attribute, positive values select part of constructed variable
Offset/Attribute		-2 147 483 648 to +2 147 483 647	APDU is a request APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND VariableIdentifier.Code.Offset/Attribute size = Integer32	Negative values select attribute, positive values select part of constructed variable
Offset/Attribute		-32 768 to +32 767	APDU is a response APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND DataFieldFormat.Offset/Attribute size= Integer16	Negative values select attribute, positive values select part of constructed variable
Offset/Attribute		-2 147 483 648 to +2 147 483 647	APDU is a response APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND DataFieldFormat.Offset/Attribute size= Integer32	Negative values select attribute, positive values select part of constructed variable
Data		Any		

Field name	Subfield name	Possible values	Constraint (present if)	Comment
RequestedLength		0 to 65 535	APDU is a request APDU AND ControlStatus.Instruction indicates Load OR Segmented Load	Indicates the length of data to Read, as the number of octets
Sequence		0 to 2	APDU is a request APDU AND ControlStatus.Instruction indicates Segmented Load OR Segmented Store	Indicates whether this request is the first, one in the middle, or the last of a segmented transfer.

4.2 Data types

The notation for data types is the same as IEC 61158-6:2003, Type 1, for the following types:

- Integer, Integer8, Integer16, Integer32
- Unsigned, Unsigned8, Unsigned16
- Floating32, Floating64

5 Transfer syntaxes

5.1 APDU encoding

5.1.1 APDU Header encoding

5.1.1.1 APDU header structure

The abstract syntax of the APDU header is defined in 4.1.2. Subclause 5.1 describes the encoding of the header. The APDU header consists of three fields, as shown in Figure 2.

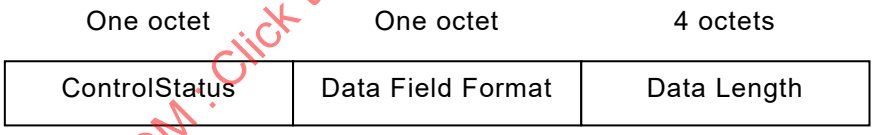


Figure 2 – APDU header structure

5.1.1.2 ControlStatus

ControlStatus is coded into one octet. If the APDU is a request APDU, the ControlStatus holds the subfields instruction, addressing method, SourceID and SecureDataExchange. The interpretation of this octet is shown in Figure 3.

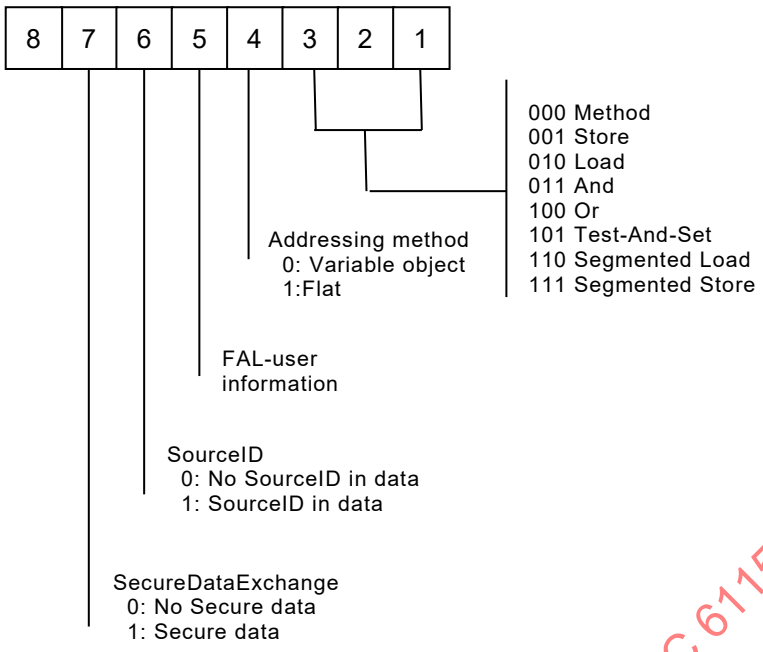


Figure 3 – Subfields of ControlStatus for Request

If the instruction is = 000 (= Errorcode), the remaining five bits of ControlStatus holds the error code. The possible values are shown in Figure 4.

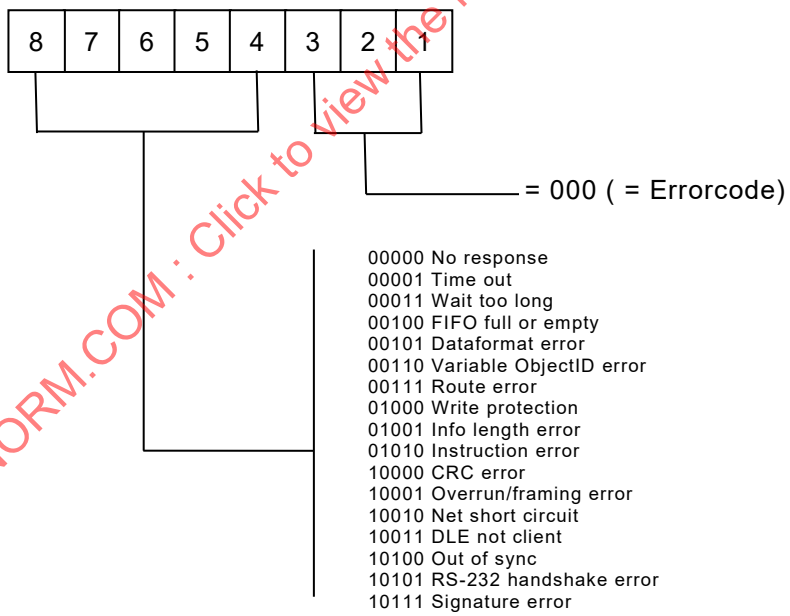


Figure 4 – Subfields of ControlStatus for Response with error

If the instruction is <> 000 (<> Errorcode), the remaining five bits of ControlStatus holds the subfields Statuscode and HistoricalDataError. The coding of these fields is shown in Figure 5.

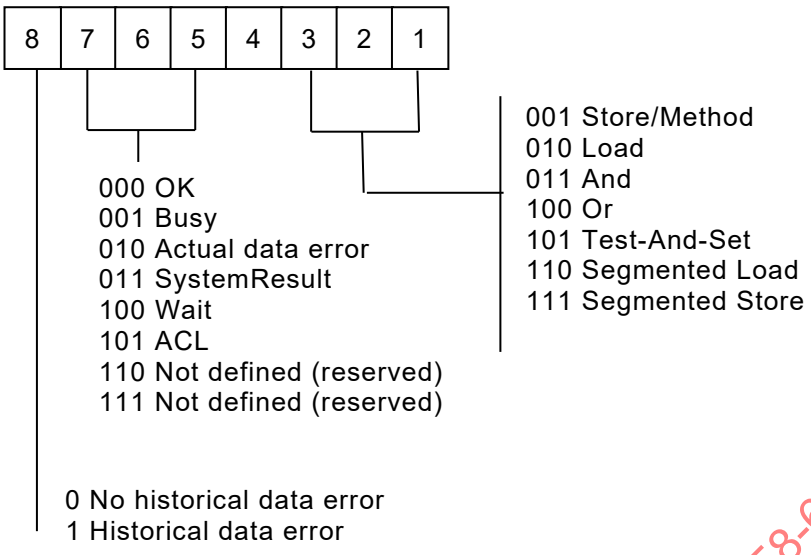


Figure 5 – Subfields of ControlStatus for Response with no error

5.1.1.3 DataFieldFormat

DataFieldFormat is coded into one octet. The coding of this octet is shown in Figure 6.

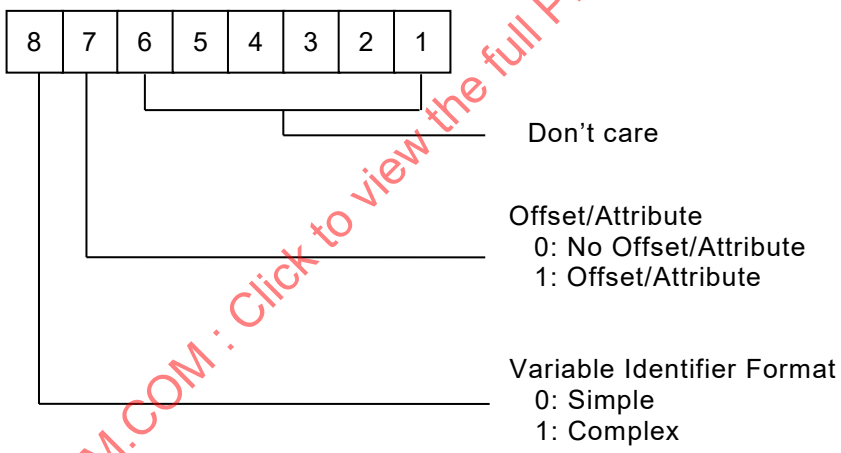


Figure 6 – DataFieldFormat encoding

5.1.1.4 DataLength

DataLength is an Integer32, indicating the total length of the APDU body.

5.1.2 APDU body encoding

5.1.2.1 APDU body structure

The abstract syntax for the APDU Body is described in 4.1.3. Subclause 5.1.2 describes the encoding. The interpretation of the APDU Body is indicated by the APDU header.

A request APDU body may consist of up to four of five possible fields, as shown in Figure 7.

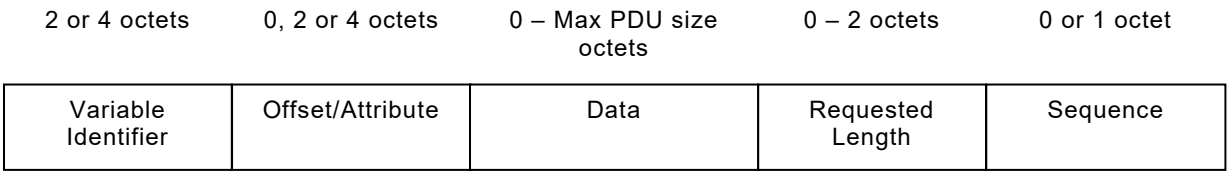


Figure 7 – Structure of request APDU body

A response APDU body may consist of up to two fields, as shown in Figure 8.

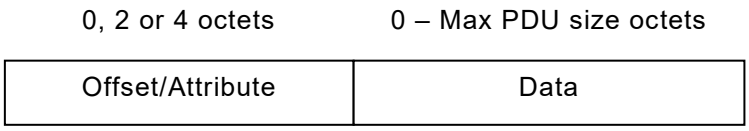


Figure 8 – Structure of response APDU body

5.1.2.2 Variable identifier

The Variable Identifier can be either simple or complex. If it is simple, it consists of only one subfield, ID, which is of type Integer16. If it is complex, it consists of two subfields, code (1 octet) and ID (3 octets) as shown in Figure 9.

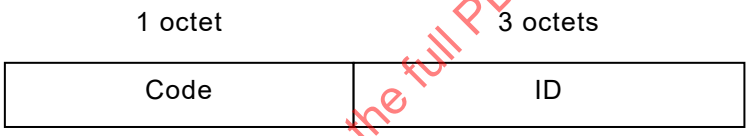


Figure 9 – Variable identifier

The coding of the Code subfield of the variable identifier is shown in Figure 10.

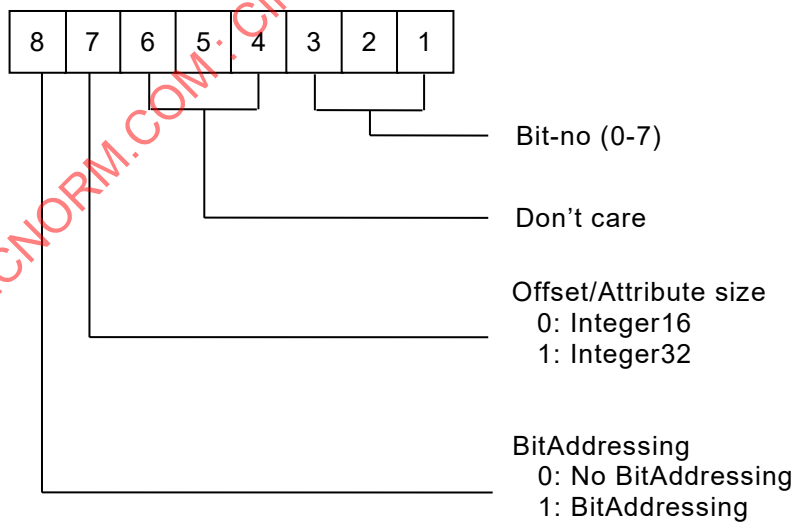


Figure 10 – Code subfield of variable identifier

5.1.2.3 Offset/attribute

The Offset/Attribute subfield of the APDU body may or may not be present. If present, Offset/Attribute is an Integer16 or an Integer32. A negative value selects an attribute of the Variable object. A positive value selects a part of the data value of the Variable object, by indicating the offset in octets to the starting octet of the data block to be transferred, relative to the first octet of the variable.

5.1.2.4 RequestedLength

The RequestedLength subfield may or may not be present.

If the APDU is a request APDU, and the ControlStatus.Instruction subfield of the APDU Header indicates Load or Segmented Load, the RequestedLength subfield is present. It indicates the octet length of the requested data or attribute.

The RequestedLength subfield is an Unsigned8 or an Unsigned16. As there is no Data subfield in the APDU if there is a RequestedLength subfield, the size of RequestedLength is implicitly given by the DataLength parameter. If the value of RequestedLength is less than 256, RequestedLength shall be of type Unsigned8.

5.1.2.5 Data

The Data subfield of the APDU body holds either:

a) Data, coded and packed as described in 5.2, or

b) an attribute of a variable object, coded and packed as described in 5.2.

As there is no RequestedLength subfield in the APDU if there is a Data subfield, the size of the data subfield is implicitly given by the DataLength parameter.

5.1.2.6 Sequence

The Sequence subfield is one octet, with the following legal values.

0: Indicating, that this is the first request of a segmented Load or Store.

1: Indicating, that this is one of the following requests of a segmented Load or Store.

2: Indicating, that this is the last request of a segmented Load or Store.

5.2 Variable object encoding and packing

5.2.1 Encoding of simple variables

5.2.1.1 Encoding of a Boolean value

a) The encoding of a boolean value shall be primitive. The ContentsOctets shall consist of a single octet.

b) If the boolean value is FALSE, bit 1 of the ContentsOctets shall be 0 (zero). If the boolean value is TRUE, bit 1 of the ContentsOctets shall be 1 (one).

5.2.1.2 Encoding of an Integer value

As defined in IEC 61158-6:2003, Type 1, Transfer syntax 1, Encoding of an Integer Value.

5.2.1.3 Encoding of an Unsigned value

As defined in IEC 61158-6:2003, Type 1, Transfer syntax 1, Encoding of an Unsigned Value, types Unsigned8 and Unsigned16.

5.2.1.4 Encoding of a Floating Point value

As defined in IEC 61158-6:2003, Type 1, Transfer syntax 1, Encoding of a Floating-Point Value.

5.2.2 Encoding of constructed variables

5.2.2.1 Encoding of a String value

a) The encoding of a variable length String value shall be primitive.

b) The Length field shall indicate as a binary number the number of elements in the String value.

c) The Length field shall be in the first octet.

5.2.2.2 Encoding of a BitString value

a) The encoding of a BitString value shall be primitive.

b) There is no Length field in the BitString.

c) The value of the BitString, commencing with the first bit and proceeding to the trailing bit, shall be placed in bits 1 to 8 of the first octet, followed by bits 1 to 8 of the second octet, followed by bits 1 to 8 of each octet up to and including the last octet of the ContentsOctets.

d) Unused bits, if any, shall be placed in bits 2-8 of the last octet.

5.2.2.3 Encoding of a Method result

a) The encoding of a Method result shall be primitive.

b) The Method result can be a simple variable or a constructed variable.

c) If the subfield Statuscode in ControlStatus is SystemResult, the length of the APDU data is 2 higher than the length of the Method result and the first 2 octets of the APDU data holds a user defined result information about the activated Method in the Variable identifier.

5.2.2.4 Structure of APDU Body subfield

The data in the APDU Body subfield is packed in a sequence as shown in Figure 11:

User data	MethodID	SourceID	Nonce	Signature

Figure 11 – Sequence of data in the APDU body subfield

MethodID is 2 or 4 octets depending on the value of Addressing method for the Method instruction.

SourceID is 2 octets and is only present in a request APDU when the SourceID subfield is SourceIDInData in Controlstatus.

Nonce is a Bitstring[64], 8 octets, and is only present in a request APDU when the SecureDataExchange subfield is SecureData in Controlstatus and the instruction is Load or Segmented Load. If Nonce is present in the APDU body, Signature is not present.

Signature is a Bitstring[64], 8 octets, and is only present in a Request APDU when the SecureDataExchange subfield is SecureData in ControlStatus. For a request APDU, Signature is only present if the instruction is Method, Store, And, Or or Segmented Store. For a Response APDU, Signature is present if SecureData was requested in SecureDataExchange. If Signature is present in the APDU body, Nonce is not present.

5.2.2.5 Calculation of Signature

Signature (Authentication code), datatype Bitstring[64], is generated as follows:

HFUNC(Key, Param, MSG)

Where HFUNC is a cryptographic hash function using a Key, datatype Bitstring[128], that is common to both the requestor and the responder. For a Read type request, Param is MSG from the received request. For a Write or Method type request, Param is the Nonce that has previously been received by the Server (also shown in the example in 8.2.2.3.1). For a Write or Method type response, Param is the Nonce that has previously been received by the requestor (also shown in the example in 8.2.2.3.1). Nonce is datatype Bitstring[64]. MSG is defined as shown in Figure 12.

HFUNC, as defined in ISO/IEC 9797-1 MAC Algorithm 1 (AES-CBC-MAC).

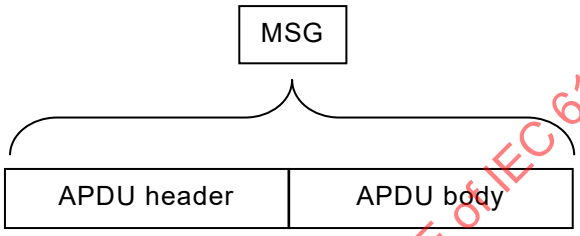


Figure 12 – MSG consists of APDU header and APDU body

5.2.3 Alignment

5.2.3.1 General

Subclause 5.2.2.3 describes how fields and elements of constructed variables are aligned and transferred.

The general alignment is two.

- Fields and elements of basic type, and with a size of 1 octet, shall be transferred immediately after the previous field or element.
- Fields and elements with a size of more that 1 octet, shall be transferred with an even offset relative to the first octet of the constructed variable.
- Fields and elements of constructed type shall be transferred with an even offset relative to the first octet of the constructed variable.

5.2.3.2 Array elements transfer syntax

Table 4 shows an example of transfer syntax for a variable "ArrayVar" of type

ARRAY[1..2] of ARRAY[1..3] of Integer8.

Table 4 – Transfer syntax for Array

1. octet	ArrayVar[1,1]
2. octet	ArrayVar[1,2]
3. octet	ArrayVar[1,3]
4. octet	ArrayVar[2,1]
5. octet	ArrayVar[2,2]
6. octet	ArrayVar[2,3]

5.2.3.3 Structure fields transfer syntax

Table 5 shows an example of transfer syntax for a variable "StructVar" of type

```
STRUCTURE
      Field1: Integer8;
      Field2: STRUCTURE
                Sub1: Integer16;
                Sub2: BitString[8];
        END;
      Field3: Integer8;
      Field4: BitString[8];
      Field5: Integer8;
END;
```

Table 5 – Transfer syntax for Structure

1. octet	StructVar.Field1
2. octet	Dummy
3. octet	StructVar.Field2.Sub1, Most significant octet
4. octet	StructVar. Field2.Sub1, Least significant octet
5. octet	StructVar.Field2.Sub2
6. octet	StructVar.Field3
7. octet	StructVar.Field4
8. octet	StructVar.Field5

5.2.4 Variable object attributes

All variable objects may have the optional attributes defined in Table 6.

Table 6 – Common variable object attributes

Attribute name	Index	Format
Variable Type Identifier	-20	Unsigned8
Octet Length	-24	Integer32
Read enable	-28	Boolean
Write enable	-29	Boolean
Write protected	-30	Boolean

The key attribute, variable identifier, is used to identify the variable object, and is not an accessible attribute.

The attributes of a variable object may be accessed from the network. If a variable object does not support access of an attribute, it shall respond to an access attempt with the errorcode "Variable Object ID error". Only one attribute can be accessed as a result of one service invocation.

There is one set of attributes for each variable object. This means, the subfields or elements of constructed variables do not have their own attributes.

Table 7 shows the variable identifier values for the variable types.

Table 7 – Variable type identifiers

Variable type	Identifier
Boolean	43
Integer8	49
Integer16	34
Integer32	35
Unsigned8	48
Unsigned16	50
Float32	36
Float64	37
UNICODE Char	51
Complex	61
String	40
BitString	62
FIFO	63

Variable objects of type FIFO shall have the mandatory additional attributes defined in Table 8.

Table 8 – FIFO variable object attributes

Attribute name	Index	Format
Next Element In	-2	Integer16
Next Element Out	-4	Integer16
Free elements	-6	Integer16
Used elements	-8	Integer16
Reread	-9	Boolean
Rewrite	-10	Boolean

5.3 Error codes

Subclause 5.3 specifies the hexadecimal values for error codes in the error status parameter of the Variable ASE service RESPONSE. The possible values are shown in Table 9.

Table 9 – Error codes

Error description	Error code (binary)
Instruction Error	01010000
Info length error	01001000
FIFO full or empty	00100000
Write protection	01000000
Data format error	00101000
Variable Object ID error	00110000
Time Out	00001000
Actual data error	x010xaaa
Historical data error	1xxxxaaa
Route error	00111000
No response	00000000
Wait too long	00011000
Out of sync	10100000
CRC error	10000000
DLE not Client	10011000
Net shortcircuit	10010000
Overrun/Framing error	10001000
RS-232 handshake error	10101000
Signature error	10111000
"x" means the bit is don't care. "aaa" means at least one of the three bits is true (1).	

6 FAL protocol state machines

Interface to FAL services and protocol machines are specified in the following paragraphs.

The behaviour of the FAL is described by three integrated protocol machines. The three protocol machines are: the FAL Service Protocol Machine (FSPM), the Application Relationship Protocol Machine (ARPM), and the Data-link Layer Mapping Protocol Machine (DMPM). The relationship among these protocol machines as well as primitives exchanged among them are depicted in Figure 13.

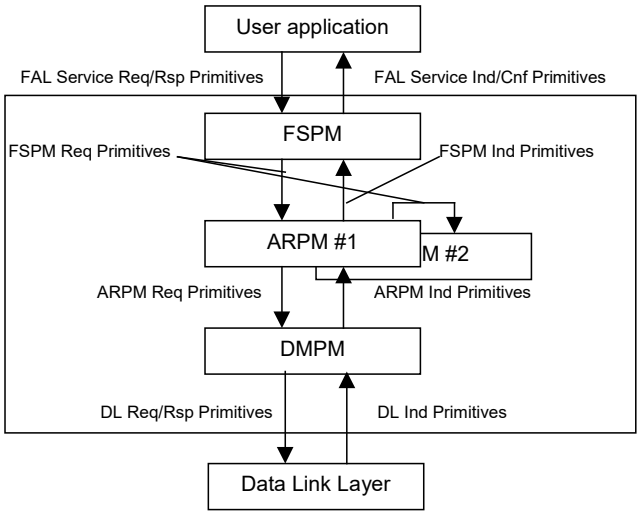


Figure 13 – Summary of FAL architecture

The FSPM describes the service interface between the FAL user and a REP. The FSPM is responsible for the following activities.

– To accept service primitives from the FAL service user and convert them into FAL internal primitives.

– To select the AREP identified by the Destination Route parameter supplied by the user application and send FAL internal primitives to the selected AREP.

– To accept FAL internal primitives from the AREP, perform the actions specified in these primitives, and return the result to the AREP from which they are received. This applies for indications holding Request APDUs.

– To accept FAL internal primitives from the AREP, convert them into service, and deliver these service primitives to the FAL user, as a result of the FAL user invocation of the RESPONSE service. This applies for indications holding Response APDUs.

The ARPM describes exchange of FAL PDUs with remote ARPMs. It does not have any state changes. The ARPM is responsible for the following activities.

– To accept FAL internal primitives from the FSPM and create and send other FAL internal primitives to the DMPM.

– To accept FAL internal primitives from the DMPM and send them to the either the FSPM as indications, or another ARPM as requests.

The DMPM describes the mapping between the FAL and the DLL. It does not have any state changes. The DMPM is responsible for the following activities.

– To accept FAL internal primitives from the ARPM, prepare DLL service primitives, and

– To receive DLL indication primitives from the DLL and send them to the ARPM in a form of FAL internal primitives.

7 AP-context state machine

The AP-Context State Machine is not present.

8 FAL service protocol machine (FSPM)

8.1 Primitives exchanged between FAL User and FSPM

The primitives exchanged between FAL user and FSPM are shown in Table 10.

Table 10 – Primitives exchanged between FAL-User and FSPM

Primitive name	Source	Associated parameters	Comments
REQUEST.req	FAL user	REP, Variable Object ID, Variable Service, Data length, Offset/Attribute, Bit-no, Data	This primitive is used to convey a FAL user request to a REP
RESPONSE.cnf	FSPM	REP, Variable Service, Data length, Error status, Data	This primitive is used to convey a response from a REP to the FAL user

8.2 FSPM states

8.2.1 General

The following states are defined for an REP: IDLE RESERVED, WAITING FOR RESPONSE, RESPONSE RECEIVED, NOT IN USE.

An REP can be in the role of proxy object (client) or real object (server). As the behaviour is very different for proxy object REPs and real object REPs, they are described in separate subclauses.

8.2.2 FSPM proxy object states

8.2.2.1 Overview

The following states apply to a REP in the proxy object role:

NOT IN USE

The REP is currently not in use. The only service primitive allowed is ReserveREP.req. All other primitives shall be rejected.

IDLE RESERVED

The REP is reserved by the FAL user, but is currently not active. Allowed service primitives in this state are REQUEST.req, Get REP Attribute.req, Set REP Attribute.req and Free REP.req. All other service primitives shall be rejected.

WAITING FOR RESPONSE

The REP has initiated a transmission, and is monitoring this by a timer. The only service primitive allowed is AR Send.ind from the ARPM, holding the response. All other service primitives shall be rejected.

RESPONSE RECEIVED

The REP has received a response (by an AR Send.ind from the ARPM), or has timed out, and is ready to deliver the response to the FAL user by a RESPONSE.cnf service primitive. Allowed service primitives in this state are REQUEST.req, Get REP Attribute.req, Set REP Attribute.req and Free REP.req from the FAL user. All other service primitives shall be rejected.

Figure 14 depicts the FSP Proxy object state machine.

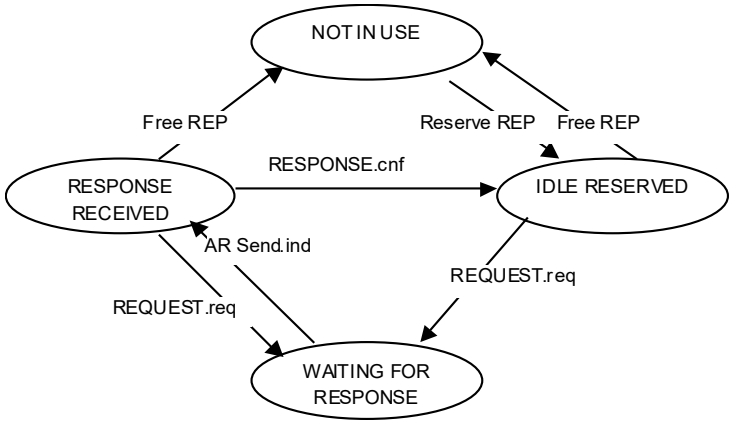


Figure 14 – FSPM proxy object state machine

8.2.2.2 Sender state transitions

8.2.2.2.1 REQUEST.req

As a result of this service invocation, the FSPM shall build APDU Header, APDU Body and Route Information, and send them to the ARPM in the form of an AR Send primitive. If anything fails, the request is rejected by returning REQUEST result FAILURE. Constraints are listed in Table 11.

Table 11 – REQUEST.req FSPM constraints

Condition ID	Condition description
1	REP Attribute State shall be IDLE RESERVED or RESPONSE RECEIVED
2	REP Attribute Role shall be Proxy object
3	If REP attribute Confirmation indicates Confirmed, no elements of REP attribute Destination Route may hold the Broadcast address (126)
4	First element of REP attribute Destination Route shall hold the address of an AREP in state OPEN
5	If parameter Data length exceeds addressed AREP attribute MaxDataSize, Variable Service may only be Read or Write
6	If parameter Variable Service is Test-And-Set, Data length shall be 1
7	If parameter Bit-no indicates bit addressing, Data length shall be 1, and parameter Variable Service shall be Read, Write or Test-And-Set

If all of the above is fulfilled, the AR Send service primitives are built as described in Table 12.

Table 12 – REQUEST.req FSPM actions

Action ID	Action description
1	The REP attribute Destination Route is copied to Route info.Destination Route
2	The REP attribute Source Route is copied to Route info.Source Route
3	The REP attribute Endpoint Address is appended to Route info.Source Route
4	If the REP attribute Confirmation indicates Unconfirmed, and no elements of Destination route holds the Broadcast node address (126), the No response node address (0) is appended to Route info.Source Route
5	The REP attribute Priority is copied to Route info.Priority
6	The REP attribute Remote LAN Server ID (if present) is copied to Route info. Remote LAN Server ID
7	Set internal variable Local Bit-no to indicate not bit-addressing, and copy parameter Data length to local variable Data length

8	Handle bit-addressing as described in the following, if parameter Bit-no indicates bit-addressing, and REP attribute Capabilities indicates bit addressing is illegal:
	Save parameter Bit-no in internal variable Local Bit-no, change parameter Bit-no to indicate not bit-addressing, and increment Local Bit-no by one
	If Variable Service parameter is Write and parameter Data.Bit1 is TRUE then
	Set APDU Header.ControlStatus.Instruction to Or, and
	clear all bits in parameter Data and set bit indicated by Local Bit-no
	If Variable Service parameter is Write and parameter Data.Bit1 is FALSE then
	Set APDU Header.ControlStatus.Instruction to And, and
	Set all bits in parameter Data and clear bit indicated by Local Bit-no
	If Variable Service parameter is And, Or or Test-And-Set then
	Set APDU Header.ControlStatus.Instruction to And, Or, Test-And-Set respectively, and
	rotate parameter Data.Bit1 to position indicated by Local Bit-no
	If Variable Service parameter is Read then
	Set APDU Header.ControlStatus.Instruction to Load

9	If parameter Data length exceeds addressed AREP attribute MaxDataSize, and parameter Variable Service is Read, set APDU Header.ControlStatus.Instruction to Segmented Load
10	If parameter Data length exceeds addressed AREP attribute MaxDataSize, and parameter Variable Service is Write, set APDU Header.ControlStatus.Instruction to Segmented Store
11	If actions 8, 9 and 10 do not apply, set then set APDU Header ControlStatus.Instruction according to the following: Parameter Variable Service Method -> Store Parameter Variable Service Write -> Store Parameter Variable Service Read -> Load Parameter Variable Service And -> And Parameter Variable Service Or -> Or Parameter Variable Service Test-And-Set -> Test-And-Set
12	If REP attribute Flat addressing indicates Flat addressing, set APDU Header ControlStatus.Addressing method to Flat. If not, set to Variable Object addressing
13	If parameter Bit-no indicates bit-addressing, or the value of parameter Variable Object ID is lower than –32768 or higher than +32767, or the value of parameter Offset/Attribute is lower than –32768 or higher than +32767, then set APDU Header DataFieldFormat.Variable Identifier Format to Complex, and set APDU Header Data length to 4. If this does not apply, then set APDU Header DataFieldFormat.Variable Identifier Format to Simple, and set APDU Header Data length to 2

Action ID	Action description
14	If the value of parameter Offset/Attribute is zero, set APDU Header DataFieldFormat.Offset/Attribute to indicate no Offset/Attribute. If not, set to indicate Offset/Attribute
15	If parameter Bit-no indicates bit-addressing, set APDU Body Variable Identifier.Code to indicate Bit-addressing, and insert Bit-no
16	If the value of parameter Offset/Attribute is lower than –32768 or higher than +32767, set APDU Body Variable Identifier.Code to indicate Integer32 Offset/Attribute size
17	If the value of parameter Offset/Attribute is not zero, and within the range of Integer16, copy lower 2 octets to APDU Body Offset/Attribute, and increment APDU Header Data length by 2. If this does not apply, copy parameter Offset/Attribute to APDU Body Offset/Attribute, and increment APDU Header Data length by 4
18	If APDU Header.ControlStatus.Instruction indicates Segmented Load, set RequestedLength in APDU Body Data first octet to "addressed AREP attribute MaxDataSize", if that value is less than or equal to 256, and set APDU Body Data first 2 octets to "addressed AREP attribute MaxDataSize", if that value is higher than 256. Set next octet of APDU Body Data to 0, indicating that this is the first request of a segmented transaction. Increment APDU Header Data length by 2 or 3, depending on size of RequestedLength. Decrement local variable Data length by "addressed AREP attribute MaxDataSize"
19	If APDU Header.ControlStatus.Instruction indicates Segmented Store, copy the first "addressed AREP attribute MaxDataSize – 2" octets from parameter Data to APDU Body Data. Set next octet of APDU Body Data to 0, indicating that this is the first request of a segmented transaction. Increment APDU Header Data length by "addressed AREP attribute MaxDataSize – 1". Decrement local variable Data length by "addressed AREP attribute MaxDataSize – 2"
20	If APDU Header.ControlStatus.Instruction indicates Load, set RequestedLength in APDU Body Data first octet to the value of parameter Data length, and increment APDU Header Data length by 1
21	If APDU Header.ControlStatus.Instruction indicates Store, And, Or or Test-And-Set, copy "parameter Data length" octets from parameter Data to APDU Body Data set, and increment APDU Header Data length by the value of "parameter Data length"
22	Send an AR Send request to the addressed AREP
23	If APDU Header.ControlStatus indicates SecureData and the ControlStatus.Instruction indicates a Read type instruction, generate a NonceForResponse and insert in the APDU Body data according to the rules defined in 5.2.2.4.
24	If APDU Header.ControlStatus.Instruction indicates SecureData and ControlStatus.Instruction indicates a Write type instruction, generate and insert SignatureForStoreRequest using NonceForStoreRequest according to the rules defined in 5.2.2.4 in the APDU Body data,
25	If the request fails, return REQUEST.cnf result FAILURE. If the request succeeds, return REQUEST.cnf result OK
26	If the request succeeds, and should be confirmed, set REP Attribute State to WAITING FOR RESPONSE
27	If the request succeeds, and should not be confirmed, and is a sequenced transaction, build the next APDU in the sequence. This continues, until the complete operation is performed, or an error occurs

8.2.2.3 Receiver state transitions

8.2.2.3.1 RESPONSE.cnf

As a result of this service invocation, the FSPM shall check, if a response has been received from the ARPM (FSPM state RESPONSE RECEIVED). If this is the case, deliver the received Variable Service, Data length, Error status and Data to the FAL user. If this is not the case, deliver the Error status "No response". Constraints are listed in Table 13.

Table 13 – RESPONSE.cnf FSPM constraints

Condition ID	Condition description
1	REP Attribute State shall be RESPONSE RECEIVED

If the above is fulfilled, the RESPONSE.cnf primitives are built as described in Table 14.

Table 14 – RESPONSE.cnf FSPM actions

Action ID	Action description
1	Copy lower 3 bits of local variable ControlStatus to parameter Variable Service
2	Copy local variable Data length to parameter Data length
3	Copy local variable ControlStatus to parameter Error status
4	Copy local variable Data to parameter Data
5	If requesting APDU Header.ControlStatus.Instruction indicates SecureData and Signature is present in the APDU Body Data according to the rules defined in 5.2.2.4, and if the APDU Header.ControlStatus.instruction is a Read type instruction, validate SignatureForResponse using NonceForResponse. Remove Signature from the APDU Body before validation. If validation fails, set ErrorCode to Signature error.
6	If requesting APDU Header.ControlStatus.Instruction indicates SecureData and Signature is present in the APDU Body Data according to the rules defined in 5.2.2.4, and if the APDU Header.ControlStatus.instruction is Method, Store, And, Or or Segmented Store, validate SignatureForStoreResponse using NonceForStoreResponse. Remove Signature from the APDU Body before validation. If validation fails, set ErrorCode to Signature error.
7	Set REP Attribute State to IDLE RESERVED

Examples of Read request/response APDU using SecureDataExchange:

Read request APDU:

APDU Header[Load, SecureData] – APDU Body[UserData, NonceForResponse]

Read response APDU:

APDU Header[Load] – APDU Body[UserData, SignatureForResponse]

Example of a Write request/response using SecureDataExchange:

The example shows a Write sequence, consisting of Nonce exchange between Requestor and Responder, followed by a Store instruction, which will result in Signature and verification in both Request and Response for the Store instruction.

Method request APDU to pass Nonce to the responder to be used for the following Store response:

APDU Header[Method, NoSecureData] – APDU Body[Parameters (=NonceForStoreResponse), MethodID]

Method response APDU, holding Nonce to be used by the requestor for the following Store request:

APDU Header[Method, NoSecureData] – APDU Body[Result(=NonceForStoreRequest)]

Write request APDU:

APDU Header[Store, SecureData] – APDU Body[UserData, SignatureForStoreRequest]

Write response APDU:

APDU Header[Store, SecureData] – APDU Body[SignatureForStoreResponse]

8.2.2.3.2 AR Send.ind

As a result of this service invocation, the FSPM shall check, if a response is expected from the ARPM (FSPM state WAITING FOR RESPONSE). If this is the case, either parse the received APDU into local variables and change REP state to RESPONSE RECIVED, or initiate the next AR Send.req if sequenced transaction. If anything fails, do nothing. Constraints are listed in Table 15.

Table 15 – AR Send.ind proxy FSPM constraints

Condition ID	Condition description
1	REP Attribute State shall be WAITING FOR RESPONSE

If the above is fulfilled, the AR Send indication is handled as described in Table 16.

Table 16 – AR Send.ind proxy FSPM actions

Action ID	Action description
1	If APDU Header.ControlStatus.Instruction indicates Sequenced Load or Sequenced Store, and this is not the final, build next APDU, and Send a new AR Send request to the addressed AREP. If Sequenced Load, copy received APDU Body Data to local variable Data.
2	If not sequenced transaction, or sequenced transaction finished, copy received APDU Body Data to local variable Data. If Load or Test-And-Set and bit-addressing, copy received APDU Body Data bit "local variable Bit-no" to bit 1 of local Variable Data. Copy received APDU Header ControlStatus to local variable ControlStatus. Copy accumulated received APDU Header Data length to local variable Data length. Set REP Attribute State to IDLE RESERVED

8.2.3 FSPM real object state machine description

8.2.3.1 Overview

The following states apply to a REP in the Real object role:

NOT IN USE

The REP is currently not in use. The only service primitive allowed is ReserveREP.req. All other service primitives shall be rejected. Typically, an REP in the Real object role will never be in this state, but will automatically go into the IDLE RESERVED state.

IDLE RESERVED

Typically the initial state of an REP in the Real object role. The REP is reserved by the FAL user, or entered this state automatically, but is not currently active. Allowed service primitives in this state for Real object role REPs are Get REP Attribute.req, Set REP Attribute.req and Free REP.req from the FAL user, and AR Send.ind from the ARPM. All other service primitives shall be rejected.

WAITING FOR RESPONSE

An REP in the Real Object role will never go into this state.

RESPONSE RECEIVED

An REP in the Real Object role will never go into this state.

Figure 15 depicts the states of the FSPM Real object state machine.

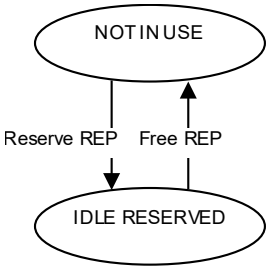


Figure 15 – FSPM real object state machine

8.2.3.2 State transitions

8.2.3.2.1 AR Send.ind

As a result of this service invocation, the FSPM shall check, if it is ready to receive an indication from the ARPM (FSPM state IDLE RESERVED). If this is the case, check the received APDU, and if OK interact directly with the addressed Variable Object, which returns the result. Finally, the FSPM shall issue an AR Send.req primitive, to deliver the result. Constraints are listed in Table 17.

Table 17 – AR Send.ind real FSPM constraints

Condition ID	Condition description
1	REP Attribute State shall be IDLE RESERVED
2	APDU Body field Variable Object Identifier shall indicate a legal Variable Object

If the above is fulfilled, the AR Send indication is handled as described in Table 18.

Table 18 – AR Send.ind real FSPM Actions

Action ID	Action description
1	If the addressed Variable Object cannot respond within the time specified by AREP Attribute MaxIndicationDelay, issue an AR Acknowledge service primitive. The Destination Route parameter shall be a copy of the Source Route parameter of the AR Send indication. The Source Route parameter shall be a copy of the Destination Route parameter of the AR Send indication. The Priority parameter shall be 0. The Confirmation parameter shall be Unconfirmed. The Remote LAN Server ID parameter shall be a copy of the Remote LAN Server ID parameter of the AR Send indication.
2	If APDU Header.ControlStatus indicates SecureData and the ControlStatus.Instruction indicates a Read type instruction, interact directly with the addressed Variable Object, which shall build the appropriate response APDU.Then generate a SignatureForResponse and insert in the APDU Body data according to the rules defined in 5.2.2.4. If APDU Header.ControlStatus indicates SecureData and the ControlStatus.Instruction indicates a Write type instruction, validate the SignatureForStoreRequest using NonceForStoreRequest, If validation fails, reject the indication and set ErrorCode to Signature error. Otherwise, interact directly with the addressed Variable Object, which shall build the appropriate response APDU.Then generate a SignatureForStoreResponse using NonceForStoreResponse and insert in the APDU Body data according to the rules defined in 5.2.2.4. If APDU Header.ControlStatus indicates No SecureData, interact directly with the addressed Variable Object, which shall build the appropriate response APDU.
3	Issue an AR Send service primitive. The Destination Route parameter shall be a copy of the Source Route parameter of the AR Send indication. The Source Route parameter shall be a copy of the Destination Route parameter of the AR Send indication. The Priority parameter shall be 0. The Confirmation parameter shall be Unconfirmed. The Remote LAN Server ID parameter shall be a copy of the Remote LAN Server ID parameter of the AR Send indication. APDU Header and APDU Body parameters shall be as build by the Variable Object.

9 Application relationship protocol machine (ARPM)

9.1 Primitives exchanged between ARPM and FSPM

The primitives exchanged between ARPM and FSPM, and between one ARPM and another are shown in Table 19 through Table 21.

Table 19 – Primitives issued by FSPM to ARPM

Primitive name	Source	Associated parameters	Comments
AR Send.req	FSPM	Route info, APDU Header, APDU Body	This primitive is used to convey an APDU holding a request or a response from the FSPM to the ARPM
AR Acknowledge.req	FSPM	Route info	This primitive is used by the FSPM to indicate, that the Variable Object is not able to handle the preceding indication immediately

Table 20 – Primitives issued by ARPM to FSPM

Primitive name	Source	Associated parameters	Comments
AR Send.ind	ARPM	Route info, APDU Header, APDU Body	This primitive is used to convey an APDU holding a request or a response from the ARPM to the FSPM

Table 21 – Primitives issued by ARPM to ARPM

Primitive name	Source	Associated parameters	Comments
AR Send.ind	ARPM	Route info, APDU Header, APDU Body	This primitive is used to convey an APDU holding a request or a response from the ARPM to another ARPM

9.2 ARPM States

9.2.1 General

The following states are defined for an AREP: OPEN, as depicted in Figure 16.

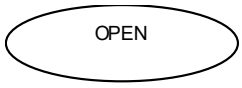


Figure 16 – ARPM state machine

OPEN

The AREP is initialised, and ready to accept service primitives.

9.2.2 Sender state transitions

9.2.2.1 AR Send.req

As a result of this service invocation, the ARPM shall deliver the received parameters to the addressed DLPM by a new AR Send service invocation. If this is not possible, the request is rejected by returning AR Send result Route error.

Constraints are listed in Table 22.

Table 22 – AR Send.req ARPM constraints

Condition ID	Condition description
1	AREP Attribute State shall be OPEN
2	First address of parameter Route info, Destination route shall specify a DLPM in state OPEN

If the above is fulfilled, the AR Send parameters are delivered to the DLPM as described in Table 23.

Table 23 – AR Send.req ARPM actions

Action ID	Action description
1	Issue an AR Send service primitive to the DLPM, with the same parameters as this indication

9.2.2.2 AR Acknowledge.req

As a result of this service invocation, the ARPM shall deliver the received parameters to the addressed DLPM by a new AR Acknowledge service invocation. If this is not possible, the request is rejected.

Constraints are listed in Table 24.

Table 24 – AR Acknowledge.req ARPM constraints

Condition ID	Condition description
1	AREP Attribute State shall be OPEN
2	First address of parameter Route info, Destination route shall specify a DLPM in state OPEN

If the above is fulfilled, the AR Acknowledge parameters are delivered to the DLPM as described in Table 25.

Table 25 – AR Acknowledge.req ARPM actions

Action ID	Action description
1	Issue an AR Acknowledge service primitive to the DLPM, with the same parameters as this indication

9.2.3 Receiver state transitions

9.2.3.1 AR Send.ind

As a result of this service invocation, the ARPM shall deliver the received parameters to the addressed destination by a new AR Send service invocation. The addressed destination may be a REP or another AREP. If it is not possible to deliver the received parameters, the request is rejected by returning AR Send result Route error.

Constraints are listed in Table 26.

Table 26 – AR Send.ind ARPM constraints

Condition ID	Condition description
1	AREP Attribute State shall be OPEN
2	First address of parameter Route info, Destination route shall either indicate an REP in state IDLE RESERVED or WAITING FOR RESPONSE, or an AREP in state OPEN

If the above is fulfilled, the AR Send parameters are delivered to the FSPM as described in Table 27.

Table 27 – AR Send.req ARPM actions

Action ID	Action description
1	If the first address of parameter Route info, Destination route indicates an REP, issue an AR Send service primitive to the addressed REP, with the same parameters as this indication
2	If the first address of parameter Route info, Destination route indicates another AREP, issue an AR Send service primitive to the addressed AREP with the same parameters as this indication, and issue an AR Acknowledge primitive to the DLPM from which this indication was received, with the Route info, Destination route parameter equal to the Route info, Source route parameter of this indication, and the Route info, Source route parameter equal to the Route info, Destination route parameter of this indication

10 DLL mapping protocol machine (DMPM)

10.1 Data-link Layer service selection

10.1.1 General

Subclause 10.1 briefly describes the Data-link Layer services utilized by the FAL. These Data-link Layer services are fully defined in the Data-link Layer service specification (IEC 61158-3-4).

10.1.2 DL-UNITDATA request

This service is used to transmit an APDU from an AREP, and deliver it as a DLSDU to a DLE.

10.1.3 DL-UNITDATA indication

This service is used to receive a DLSPDU from a DLE and deliver it as an APDU to an AREP.

10.1.4 DL-UNITDATA response

This service is used to inform the DLE, that a response will not be prepared in time.

10.1.5 DLM-Set primitive and parameters

This service is used to update the values of attributes of the DLE locally.

10.1.6 DLM-Get primitive and parameters

This service is used to read the values of attributes of the DLE locally.

10.2 Primitives exchanged between ARPM and DLPM

The primitives exchanged between DLPM and ARPM are shown in Table 28 and Table 29.

Table 28 – Primitives issued by ARPM to DLPM

Primitive name	Source	Associated parameters	Comments
AR Send.req	ARPM	Route info, APDU Header, APDU Body	This primitive is used to convey an APDU holding a request or a response from the ARPM to the DLPM
AR Acknowledge.req	ARPM	Route info	This primitive is used by the ARPM to indicate, that the Variable Object is not able to handle the preceding indication immediately

Table 29 – Primitives issued by DLPM to ARPM

Primitive name	Source	Associated parameters	Comments
AR Send.ind	ARPM	Route info, APDU Header, APDU Body	This primitive is used to convey an APDU holding a request or a response from the DLPM to the ARPM

10.3 Primitives exchanged between DLPM and data-link layer

The primitives exchanged between DLPM and ARPM are shown in Table 30 and Table 31.

Table 30 – Primitives issued by DLPM to data-link layer

Primitive name	Source	Associated parameters	Comments
DL-UNITDATA request	DLPM	Destination-DL-route, Source-DL-route, Priority, Maximum retry time, Control-status, Data-field-format, DLSDU	This primitive is used to convey a DLSDU holding a request or a response to the DLE
DL-UNITDATA response	DLPM	Destination-DL-route, Source-DL-route	This primitive is used to indicate to the DLE, that the preceding indication cannot be handled immediately

Table 31 – Primitives issued by data-link layer to DLPM

Primitive name	Source	Associated parameters	Comments
DL-UNITDATA indication	DLL	Destination-DL-route, Source-DL-route, Confirmation expected, Control-status, Data-field-format, DLSDU	This primitive is used to convey a DLSDU holding a request or a response from the DLE to the DLPM

10.4 DLPM states

10.4.1 States

The following states are defined for a DLPM: OPEN, as depicted in Figure 17.

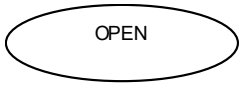


Figure 17 – DLPM state machine

OPEN

The DLPM is initialised, and ready to accept service primitives.

10.4.2 Sender state transitions

10.4.2.1 AR Send.req

As a result of this service invocation, the DLPM shall deliver the received parameters to the DLE by a DL-UNITDATA request primitive. If this is not possible, the request is rejected.

Constraints are listed in Table 32.

Table 32 – AR Send.req DLPM constraints

Condition ID	Condition description
1	DLPM state shall be OPEN

If the above is fulfilled, the AR Send parameters are converted and delivered to the DLE as described in Table 33.

Table 33 – AR Send.req DLPM actions

Action ID	Action description
1	First address of parameter Route info, Destination Route, is removed. The result is delivered as the Destination-DL-route parameter of the DL-UNITDATA request primitive
2	Parameter Route info, Source Route is delivered as the Source-DL-route parameter of the DL-UNITDATA request primitive
3	Parameter Route info, Priority is delivered as the Priority parameter of the DL-UNITDATA request primitive
4	AREP attribute MaxRetryTime is delivered as the Maximum retry time parameter of the DL-UNITDATA request primitive
5	Parameter APDU Header, ControlStatus is delivered as the Control-status parameter of the DL-UNITDATA request primitive
6	Parameter APDU Header, DataFieldFormat is delivered in bits 7 and 8, and APDU Header, DataLength is delivered in bits 1 to 6 of the Data-field-format parameter of the DL-UNITDATA request primitive
7	Parameter APDU Body is delivered as the DLSDU parameter of the DL-UNITDATA request primitive

10.4.2.2 AR Acknowledge.req

As a result of this service invocation, the DLPM shall deliver the received parameters to the DLE by a DL-UNITDATA response primitive. If this is not possible, no action is taken.

Constraints are listed in Table 34.

Table 34 – AR Acknowledge.req DLPM constraints

Condition ID	Condition description
1	DLPM state shall be OPEN

If the above is fulfilled, the AR Acknowledge parameters are converted and delivered to the DLE as described in Table 35.

Table 35 – AR Acknowledge.req DLPM actions

Action ID	Action description
1	First address of parameter Route info, Destination Route, is removed. The result is delivered as the Destination-DL-route parameter of the DL-UNITDATA response primitive
2	Parameter Route info, Source Route is delivered as the Source-DL-route parameter of the DL-UNITDATA response primitive

10.4.3 Receiver state transitions

10.4.3.1 DL-UNITDATA indication

As a result of a DL-UNITDATA indication service invocation, the DLPM shall deliver the parameters received from the DLE to the AREP by an AR Send service invocation. If this is not possible, the request is rejected.

Constraints are listed in Table 36.

Table 36 – DL-UNITDATA.ind DLPM constraints

Condition ID	Condition description
1	AREP Attribute State shall be OPEN

If the above is fulfilled, the AR Send parameters are delivered to the ARPM as described in Table 37.

Table 37 – DL-UNITDATA.ind DLPM actions

Action ID	Action description
1	AREP address is inserted in front of all elements in the parameter Source-DL-route. The result is delivered as the Route info, Source route parameter of the AR Send primitive
2	Parameter Destination-DL-route is delivered as the Route info, Source route parameter of the AR Send primitive
3	Parameter Route info, Priority is set to 0
5	Parameter Confirmation Expected is delivered as the Confirmation parameter of the AR Send primitive
6	Parameter Control-status is delivered as the APDU Header, ControlStatus parameter of the AR Send primitive
7	Bits 1 to 6 of the Data-field-format parameter are delivered as the APDU Header, DataLength parameter of the AR Send primitive
8	Bits 7 and 8 of the Data-field-format parameter are delivered as the APDU Header, DataFieldFormat parameter of the AR Send primitive
9	Parameter DLSDU is delivered as the APDU Body parameter of the AR Send primitive

11 Protocol options

There are no options to select for Type 4.

Bibliography

IEC 61158-1:2019, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-2, *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-4-4:2019, *Industrial communication networks – Fieldbus specifications – Part 4-4: Data-link layer protocol specification – Type 4 elements*

IEC 61784-1:2019, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61784-2:2019, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC/IEEE 8802-3*

SOMMAIRE

AVANT-PROPOS 45
INTRODUCTION 47
1 Domaine d'application 48
1.1 Généralités 48
1.2 Spécifications 48
1.3 Conformité 49
2 Références normatives 49
3 Termes, définitions, symboles, abréviations et conventions 49
3.1 Termes et définitions référencés 50
3.1.1 Termes de l'ISO/IEC 7498-1 50
3.1.2 Termes de l'ISO/IEC 8822 50
3.1.3 Termes de l'ISO/IEC 9545 50
3.1.4 Termes de l'ISO/IEC 8824-1 50
3.1.5 Termes relatifs à la couche Liaison de données de bus de terrain 50
3.2 Abréviations et symboles 51
3.3 Conventions 51
3.3.1 Concept général 51
3.3.2 Conventions relatives aux diagrammes d'états pour les éléments de type 4 52
4 Description de la syntaxe de la couche FAL 53
4.1 Syntaxe abstraite des unités PDU FAL-AR 53
4.1.1 Généralités 53
4.1.2 Syntaxe abstraite de l'en-tête des unités APDU 53
4.1.3 Syntaxe abstraite du corps des unités APDU 55
4.2 Types de données 56
5 Syntaxes de transfert 57
5.1 Encodage des unités APDU 57
5.1.1 Encodage de l'En-tête des unités APDU 57
5.1.2 Encodage du corps des unités APDU 59
5.2 Encodage et compression des objets de variable 61
5.2.1 Encodage de variables simples 61
5.2.2 Encodage de variables construites 61
5.2.3 Alignement 63
5.2.4 Attributs d'objets de variable 64
5.3 Codes d'erreur 65
6 Diagrammes d'états de protocole de la couche FAL 66
7 Diagramme d'états de contexte AP 68
8 Machine de protocole de service FAL (FSPM) 68
8.1 Primitives échangées entre l'utilisateur FAL et la machine FSPM 68
8.2 Etats FSPM 68
8.2.1 Généralités 68
8.2.2 Etats des objets proxy de la machine FSPM 68
8.2.3 Description du diagramme d'états de l'objet réel de la machine FSPM 74
9 Machine de protocole de relations AR (ARPM) 75
9.1 Primitives échangées entre les machines ARPM et FSPM 75
9.2 Etats de la machine ARPM 76
9.2.1 Généralités 76

9.2.2 Passages d'état de l'expéditeur 77
9.2.3 Transitions d'états du destinataire 78
10 Machine de protocole de mapping de couche DLL 78
10.1 Sélection des services de couche liaison de données 78
10.1.1 Généralités 78
10.1.2 Request de DL-UNITDATA 78
10.1.3 Indication de DL-UNITDATA 78
10.1.4 Response de DL-UNITDATA 79
10.1.5 Primitive et paramètres DLM-Set 79
10.1.6 Primitive et paramètres DLM-Get 79
10.2 Primitives échangées entre les machines ARPM et FSPM 79
10.3 Primitives échangées entre la machine DLPM et la couche Liaison de données 79
10.4 Etats de la machine DLPM 80
10.4.1 Etats 80
10.4.2 Passages d'état de l'expéditeur 80
10.4.3 Transitions d'états du destinataire 81
11 Options de protocole 82
Bibliographie 83

Figure 1 – Diagramme de passages d'état 52
Figure 2 – Structure de l'en-tête des unités APDU 57
Figure 3 – Sous-champs de ControlStatus pour une demande 57
Figure 4 – Sous-champs de ControlStatus pour une réponse avec erreur 58
Figure 5 – Sous-champs de ControlStatus pour une réponse sans erreur 58
Figure 6 – Codage de DataFieldFormat 59
Figure 7 – Structure du corps des unités APDU de demande 59
Figure 8 – Structure du corps des unités APDU de réponse 59
Figure 9 – Identificateur de variable 60
Figure 10 – Sous-trame Code de l'identificateur de variable 60
Figure 11 – Séquence de données dans le sous-champ du corps d'unité APDU 62
Figure 12 – MSG se compose de l'en-tête d'unité APDU et du corps d'unité APDU 63
Figure 13 – Résumé de l'architecture FAL 67
Figure 14 – Diagramme d'états des objets proxy de la machine FSPM 69
Figure 15 – Diagramme d'états des objets réels FSPM 74
Figure 16 – Diagramme d'états ARPM 76
Figure 17 – Diagramme d'états de la machine DLPM 80

Tableau 1 – Eléments de la description d'un diagramme d'états 52
Tableau 2 – En-tête d'unité APDU 53
Tableau 3 – Corps d'unité APDU 55
Tableau 4 – Syntaxe de transfert des matrices 64
Tableau 5 – Syntaxe de transfert de structure 64
Tableau 6 – Attributs communs des objets de variable 64
Tableau 7 – Identificateurs des différents types de variables 65
Tableau 8 – Attributs d'objets de variable de type FIFO 65

Tableau 9 – Codes d'erreur 66
Tableau 10 – Primitives échangées entre l'utilisateur FAL et la machine FSPM 68
Tableau 11 – Contraintes relatives à REQUEST.req FSPM 70
Tableau 12 – Actions relatives à REQUEST.req FSPM 70
Tableau 13 – Contraintes relatives à RESPONSE.cnf FSPM 72
Tableau 14 – Actions relatives à RESPONSE.cnf FSPM 72
Tableau 15 – Contraintes relatives à AR Send.ind proxy FSPM 73
Tableau 16 – Actions relatives à AR Send.ind proxy FSPM 74
Tableau 17 – Contraintes relatives à AR Send.ind real FSPM 75
Tableau 18 – Actions relatives à AR Send.ind real FSPM 75
Tableau 19 – Primitives adressées par la machine de protocole FSPM à la machine ARPM 76
Tableau 20 – Primitives adressées par la machine de protocole ARPM à la machine FSPM 76
Tableau 21 – Primitives adressées par une machine ARPM à une autre 76
Tableau 22 – Contraintes relatives à AR Send.req ARPM 77
Tableau 23 – Actions relatives à AR Send.req ARPM 77
Tableau 24 – Contraintes relatives à AR Acknowledge.req ARPM 77
Tableau 25 – Actions relatives à AR Acknowledge.req ARPM 77
Tableau 26 – Contraintes relatives à AR Send.ind ARPM 78
Tableau 27 – Actions relatives à AR Send.req ARPM 78
Tableau 28 – Primitives adressées par la machine ARPM à la machine DLPM 79
Tableau 29 – Primitives adressées par la machine DLPM à la machine ARPM 79
Tableau 30 – Primitives adressées par la machine DLPM à la couche Liaison de données 79
Tableau 31 – Primitives adressées par la couche Liaison de données à la machine DLPM 80
Tableau 32 – Contraintes relatives à AR Send.req DLPM 80
Tableau 33 – Actions relatives à AR Send.req DLPM 81
Tableau 34 – Contraintes relatives à AR Acknowledge.req DLPM 81
Tableau 35 – Actions relatives à AR Acknowledge.req DLPM 81
Tableau 36 – Contraintes relatives à DL-UNITDATA.ind DLPM 82
Tableau 37 – Contraintes relatives à DL-UNITDATA.ind DLPM 82

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-4: Spécification du protocole de la couche application – Eléments de type 4

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC 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, l'IEC – 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 l'IEC"). 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 l'IEC, participent également aux travaux. L'IEC 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 l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC 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 l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC 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 l'IEC. L'IEC 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 l'IEC, 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 l'IEC, 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 l'IEC ou de toute autre Publication de l'IEC, 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 l'IEC peuvent faire l'objet de droits de brevet. L'IEC 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 l'IEC 61784-1 et l'IEC 61784-2.

La Norme internationale IEC 61158-6-4 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels.

Cette troisième édition annule et remplace la deuxième édition parue en 2014. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

a) ajout de paramètres utilisateur supplémentaires aux services;
b) ajout de services supplémentaires pour prendre en charge les objets répartis;
c) ajout de services de sécurité supplémentaires.

La présente version bilingue (2021-04) correspond à la version anglaise monolingue publiée en 2019-06.

La version française de cette norme n'a pas été soumise au vote.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 61158, publiées sous le titre général *Réseaux de communication industriels – Spécifications des bus de terrain* peut être consultée sur le site web de l'IEC.

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 l'IEC sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

INTRODUCTION

Le présent document s'inscrit dans une série créée pour faciliter l'interconnexion des composants de systèmes d'automatisation. Elle renvoie aux autres normes de l'ensemble défini par le modèle de référence de bus de terrain "à trois couches" décrit dans l'IEC 61158-1.

Le protocole d'application fournit le service d'application au moyen des services disponibles au niveau de la couche Liaison de données ou de la couche immédiatement inférieure. Le principal objectif du présent document est de définir un ensemble de règles de communication, exprimées en ce qui concerne les procédures que doivent suivre les entités d'application (Application Entity, AE) homologues au moment de la communication. Ces règles de communication ont pour vocation de fournir une base de développement stable visant à atteindre différents objectifs:

- en tant que guide pour les développeurs et les concepteurs;
- réaliser les essais et acquérir l'équipement;
- dans un accord d'intégration des systèmes dans l'environnement de systèmes ouverts;
- dans le cadre d'une meilleure compréhension des communications à contrainte de temps au sein de l'OSI.

Le présent document porte en particulier sur la communication et l'interfonctionnement des capteurs, des effecteurs et d'autres appareils d'automatisation. Grâce au présent document associé à d'autres normes des modèles de référence OSI ou de bus de terrain, des systèmes par ailleurs incompatibles peuvent fonctionner ensemble, quelle que soit leur combinaison.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-4: Spécification du protocole de la couche application – Eléments de type 4

1 Domaine d'application

1.1 Généralités

La couche Application de bus de terrain (Fieldbus Application Layer, FAL) procure aux programmes de l'utilisateur un moyen d'accès à l'environnement de communication des bus de terrain. A cet égard, la FAL peut être considérée comme une "fenêtre entre programmes d'application correspondants".

La présente partie de l'IEC 61158 fournit des éléments communs pour les communications à temps critique ou non entre des programmes d'application dans un environnement et avec un matériel d'automation spécifiques aux bus de terrain de type 4. Le terme "en temps critique" signale l'existence d'une fenêtre temporelle dans laquelle des actions spécifiées doivent être exécutées, avec un niveau de certitude défini. La non-réalisation des actions spécifiées dans la fenêtre temporelle induit un risque de défaillance des applications qui demandent ces actions, avec les risques relatifs à l'équipement, les installations et éventuellement la vie humaine.

La présente norme internationale spécifie les interactions entre les applications distantes et définit le comportement, visible par un observateur externe, assuré par la couche Application de bus de terrain de type 4, en termes

a) de syntaxe abstraite formelle définissant les unités de données de protocole de couche Application, acheminées entre les entités d'application en communication;

b) de syntaxe de transfert définissant les règles de codage qui s'appliquent aux unités de données de protocole de couche Application;

c) de diagramme d'états de contexte d'application définissant le comportement de service d'application observable entre les entités d'application en communication;

d) de diagrammes d'états de relations d'applications définissant le comportement de communication visible entre les entités d'application en communication.

Le présent document vise à définir le protocole mis en place pour

1) définir la représentation filaire des primitives de service définies dans l'IEC 61158-5-4, et

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

Le présent document spécifie le protocole de la couche Application de bus de terrain de type 4, en conformité avec le modèle de référence de base OSI (ISO/IEC 7498-1) et avec la structure de la couche Application OSI (ISO/IEC 9545).

1.2 Spécifications

Le présent document a pour objectif principal de spécifier la syntaxe et le comportement du protocole de couche Application qui véhicule les services de couche Application définis dans l'IEC 61158-5-4.

Un objectif secondaire consiste à fournir des voies d'évolution à partir des protocoles de communication industriels antérieurs. Ce dernier objectif explique la diversité des protocoles normalisés dans la série IEC 61158-6.

1.3 Conformité

Le présent document ne définit pas de mises en œuvre, ni de produits particuliers, pas plus qu'elle ne limite les mises en œuvre des entités de couche Application dans les systèmes d'automatisation industriels. La conformité est obtenue par le biais de la mise en œuvre de cette spécification du protocole de la couche application.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. 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 IEC 61158, ainsi que l'IEC 61784-1 et l'IEC 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.

IEC 61158-3-4:2019, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-4: Définition des services de couche liaison de données – Eléments de type 4*

IEC 61158-5-4:2019, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-4: Définition des services de la couche application – Eléments de type 4*

ISO/IEC 7498-1, *Technologies de l'information – Modèle de référence de base pour l'interconnexion des systèmes ouverts (OSI): Le modèle de base*

ISO/IEC 8822, *Technologies de l'information – Interconnexion de systèmes ouverts – Définition du service de présentation*

ISO/IEC 8824-1, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1): Spécification de la notation de base*

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

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

ISO/IEC 9797-1, *Technologies de l'information – Techniques de sécurité – Codes d'authentification de message (MAC) – Partie 1: Mécanismes utilisant un chiffrement par blocs*

3 Termes, définitions, symboles, abréviations et conventions

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

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

3.1 Termes et définitions référencés

3.1.1 Termes de l'ISO/IEC 7498-1

Pour les besoins du présent document, les termes suivants, définis dans l'ISO/IEC 7498-1, s'appliquent:

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

3.1.2 Termes de l'ISO/IEC 8822

Pour les besoins du présent document, les termes suivants, définis dans l'ISO/IEC 8822, s'appliquent:

a) syntaxe abstraite
b) contexte de présentation

3.1.3 Termes de l'ISO/IEC 9545

Pour les besoins du présent document, les termes suivants, définis dans l'ISO/IEC 9545, s'appliquent:

a) application-association (association d'applications)
b) application-context (contexte d'application)
c) nom de contexte d'application
d) application-entity-invocation (invocation d'entité d'application)
e) application-entity-type (type d'entité d'application)
f) application-process-invocation (invocation de processus d'application)
g) application-process-type (type de processus d'application)
h) application-service-element (élément de service d'application)
i) élément de service de contrôle d'application

3.1.4 Termes de l'ISO/IEC 8824-1

Pour les besoins du présent document, les termes suivants, définis dans l'ISO/IEC 8824-1, s'appliquent:

a) identificateur d'objet
b) type

3.1.5 Termes relatifs à la couche Liaison de données de bus de terrain

Pour les besoins du présent document, les termes suivants, définis dans les normes IEC 61158-3-4 et IEC 61158-4-4, s'appliquent:

a) délai DL
b) politique de planification DL

c) DLCEP
d) DLC
e) mode orienté connexion DL
f) DLPDU
g) DLSDU
h) DLSAP
i) adresse réseau
j) adresse de nœud
k) node

3.2 Abréviations et symboles

AE	Entité d'application
AL	Couche application
ALE	Entité de la couche Application
APDU	Unité de données de protocole d'application
AR	Relation entre applications
AREP	Point de fin de relation entre applications
ASE	Elément de service d'application
Cnf	Confirmation
DL-	(comme préfixe) Liaison de données-
DLCEP	Point de fin de connexion de couche Liaison de données
DLL	Couche Liaison de données
DLE	Entité de la couche Liaison de données
DLM	Gestion de liaison de données
DLS	Service de la couche Liaison de données
DLSAP	Point d'accès de service de la couche Liaison de données
DLSDU	Unité de données de service de liaison de données
FME	Entité de gestion de la couche Application de bus de terrain
Ind	Indication
IP	Internet Protocol
PDU	Unité de données de protocole
Req	Demande
Rsp	Réponse
SME	Entité de gestion système
.cnf	Primitive de confirmation
.ind	Primitive d'indication
.req	Primitive de demande
.rsp	Primitive de réponse

3.3 Conventions

3.3.1 Concept général

La couche FAL se compose d'un ensemble d'ASE orientés objet. Chaque ASE est spécifié dans un paragraphe distinct. Chaque spécification d'ASE est divisée en trois parties: ses définitions de classe, ses services et sa spécification de protocole. Les deux premiers éléments figurent dans l'IEC 61158-5-4. La spécification des protocoles pour chacun des éléments ASE est définie dans le présent document.

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 qui utilise les services ASE de gestion spécifiés dans l'IEC 61158-5-4. La spécification de service définit les services fournis par l'élément ASE.

Le présent document emploie les conventions de description énoncées dans l'ISO/IEC 10731.

3.3.2 Conventions relatives aux diagrammes d'états pour les éléments de type 4

Un diagramme d'états décrit la séquence d'états par lesquels passe une entité; il peut se matérialiser par un diagramme de passages d'état et/ou une table d'états.

Dans un diagramme de passage d'états (Figure 1), les passages d'états sont représentés par des cercles et sont représentés par une flèche à côté de laquelle sont indiqués les événements ou conditions sous-jacents.

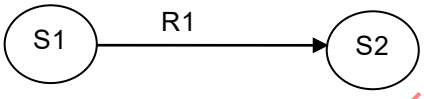


Figure 1 – Diagramme de passages d'état

Tableau 1 – Eléments de la description d'un diagramme d'états

#	Etat actuel	Evénements ou conditions déclenchant ce passage d'état => action	Etat suivant
Nom du passage	Etat actuel dans lequel se trouve le diagramme d'états lors de ce passage d'état	Evénements ou conditions déclenchant ce passage d'état. => Actions effectuées lorsque les événements ou conditions ci-dessus sont réunis. Elles figurent toujours au-dessous des événements ou conditions et sont toujours présentées avec un retrait.	Etat suivant dans lequel passe le diagramme d'états une fois les actions effectuées

Les conventions utilisées dans le tableau des transitions d'état (Tableau 1) sont les suivantes.

:= la valeur d'un élément, à gauche, est remplacée par la valeur d'un élément, à droite. Si un élément à droite est un paramètre, il provient de la primitive indiquée comme événement d'entrée.

xxx indique un nom de paramètre.

Exemple:

Identifier:= reason
signifie que la valeur d'un paramètre "reason" est attribuée à un paramètre appelé "Identifier".

"xxx" indique une valeur fixe.

Exemple:

Identifier:= "abc"
signifie que la valeur "abc" est attribuée à un paramètre appelé "Identifier".

= condition logique indiquant qu'un élément à gauche est égal à un élément à droite.
< condition logique indiquant qu'un élément à gauche est inférieur à l'élément à droite.
> condition logique indiquant qu'un élément à gauche est supérieur à l'élément à droite.
<> condition logique indiquant qu'un élément à gauche est différent d'un élément à droite.
&& indique le "ET" logique
|| indique le "OU" logique

Service.req représente une primitive de demande; Service.req{} signifie qu'une primitive de demande est envoyée;

Service.ind représente une primitive d'indication; Service.ind{} signifie qu'une primitive d'indication est reçue;

Service.rsp représente une primitive de réponse; Service.rsp{} signifie qu'une primitive de réponse est envoyée;

Service.cnf représente une primitive de confirmation; Service.cnf{} signifie qu'une primitive de confirmation est reçue.

4 Description de la syntaxe de la couche FAL

4.1 Syntaxe abstraite des unités PDU FAL-AR

4.1.1 Généralités

Les informations stockées dans une unité APDU varient selon que cette unité contient une demande ou une réponse. Le rôle du diagramme d'états qui code l'unité APDU (la machine FSPM) est de déterminer la manière dont l'unité APDU est codée.

Les unités APDU sont toujours composées d'un en-tête d'unité APDU et d'un corps d'unité APDU. Dans les unités APDU de réponse, le corps d'unité APDU peut être vide.

4.1.2 Syntaxe abstraite de l'en-tête des unités APDU

Le Tableau 2 définit le contenu de l'en-tête d'unité APDU.

Tableau 2 – En-tête d'unité APDU

Nom de champ	Nom de la sous-trame	Valeurs possibles	Contrainte (le cas échéant)	Commentaire
ControlStatus	Instruction	Errorcode Méthode Enregistrement Chargement Et Ou Soumettre à essai et définir Chargement segmenté Enregistrement segmenté		

Nom de champ	Nom de la sous-trame	Valeurs possibles	Contrainte (le cas échéant)	Commentaire
ControlStatus	Errorcode	Décrites de la Figure 3 à la Figure 5	ControlStatus.Instruction = Errorcode	
ControlStatus	Méthode d'adressage	Objet Variable Plat	ControlStatus.Instruction <> Errorcode	
ControlStatus	Méthode d'adressage	Objet Variable Plat	ControlStatus.Instruction =Method	Objet de variable qui signifie 2 octets MethodID Flat qui signifie 4 octets MethodID
ControlStatus	SourceID	NoSourceIDInData SourceIDInData		Utilisé par l'application demandeuse pour indiquer à l'application répondeuse que l'instruction exécutée peut être liée à un SourceID spécifique
ControlStatus	SecureDataExchange	NoSecureData SecureData	L'unité APDU est une unité APDU de demande	Instruction de lecture de type: Utilisé par l'application demandeuse pour assurer une réponse authentifiée. Instruction de type d'écriture: Utilisé par l'application demandeuse pour assurer une instruction de type d'écriture authentifiée dans le répondeur.
ControlStatus	ActualDataError	NoActualError ActualError	ControlStatus.Instruction <> Errorcode	Utilisé par l'application de l'utilisateur répondeur pour indiquer qu'une erreur réelle peut affecter l'objet de variable accessible
ControlStatus	SystemResult	NoSystemResult SystemResult	ControlStatus.Instruction = Méthode	Utilisé par l'application de l'utilisateur répondeur pour indiquer que des données de 2 octets supplémentaires sont insérées dans les données d'APDU avant les données de résultat relatives à l'identificateur de variable accessible Par conséquent, la longueur de DataLength est de 2 fois supérieure à la longueur de résultat de méthode.
ControlStatus	HistoricalDataError	NoHistoricalError HistoricalError	ControlStatus.Instruction <> Errorcode	Utilisé par l'application de l'utilisateur répondeur pour indiquer qu'une erreur peut avoir affecté l'objet de variable accessible
DataFieldFormat	Décalage/attribut	Pas de décalage/attribut Décalage/attribut		Indique si le corps d'unité APDU contient un champ Offset/Attribute
DataFieldFormat	Format d'identificateur de variable	Simple Complexe	L'unité APDU est une unité APDU de demande	Indique le format de l'identificateur de variable dans une unité APDU de demande

Nom de champ	Nom de la sous-trame	Valeurs possibles	Contrainte (le cas échéant)	Commentaire
DataFieldFormat	Offset/Attribute size	Integer16 Integer32	L'unité APDU est une unité de réponse APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute	Indique la taille du champ Offset/Attribute du corps d'unité APDU
DataLength		min. 2		Indique la longueur totale du corps d'unité APDU. MaxDataSize indique la longueur maximale de la partie de données du corps d'unité APDU.

4.1.3 Syntaxe abstraite du corps des unités APDU

L'en-tête d'unité APDU indique l'interprétation du contenu du corps d'unité APDU.

Le Tableau 3 définit le contenu du corps d'unité APDU.

Tableau 3 – Corps d'unité APDU

Nom de champ	Nom de la sous-trame	Valeurs possibles	Contrainte (le cas échéant)	Commentaire
VariableIdentifier	Code.Bitaddressing	No BitAddressing BitAddressing	L'unité APDU est une unité APDU de demande ET l'en-tête d'unité APDU indique Complex VariableIdentifier	Si ce champ indique BitAddressing, le VariableIdentifier contient également un Bit-no
VariableIdentifier	Code.Bit-no	0 à7	L'unité APDU est une unité de demande APDU AND APDU qui indique Complex VariableIdentifier AND VariableIdentifier indique BitAddressing	Bit-no sélectionne un bit dans un octet. Bit-no = 0 sélectionne le bit 1, etc. L'octet est sélectionné par Offset/Attribute.
VariableIdentifier	Code.Offset/Attribute size	Integer16 Integer32	L'unité APDU est une unité APDU de demande ET DataFieldFormat.Variable Identifier Format = Complex AND DataFieldFormat.Offset/Attribute = Offset/Attribute	
VariableIdentifier	ID	-32 768 à +32 767	L'unité APDU est une unité APDU de demande ET DataFieldFormat.Variable Identifier Format = Simple	
VariableIdentifier	ID	-8 388 608 à +8 388 607	L'unité APDU est une unité APDU de demande ET DataFieldFormat.Variable Identifier Format = Complex	

Nom de champ	Nom de la sous-trame	Valeurs possibles	Contrainte (le cas échéant)	Commentaire
Décalage/attribut		-32 768 à +32 767	L'unité APDU est une unité APDU de demande AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND VariableIdentifier.Code.Offset/Attribute size = Integer16	Les valeurs négatives sélectionnent un attribut, les valeurs positives sélectionnent une partie de la variable construite
Décalage/attribut		-2 147 483 648 à +2 147 483 647	L'unité APDU est une unité APDU de demande AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND VariableIdentifier.Code.Offset/Attribute size = Integer32	Les valeurs négatives sélectionnent un attribut, les valeurs positives sélectionnent une partie de la variable construite
Décalage/attribut		-32 768 à +32 767	L'unité APDU est une unité de réponse APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND DataFieldFormat.Offset/Attribute size= Integer16	Les valeurs négatives sélectionnent un attribut, les valeurs positives sélectionnent une partie de la variable construite
Décalage/attribut		-2 147 483 648 à +2 147 483 647	L'unité APDU est une unité de réponse APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND DataFieldFormat.Offset/Attribute size= Integer32	Les valeurs négatives sélectionnent un attribut, les valeurs positives sélectionnent une partie de la variable construite
Données		Tous		
RequestedLength		0 à 65 535	L'APDU est une demande APDU AND ControlStatus.Instruction indique Chargement OU Chargement segmenté	Indique la longueur des données à Lire, ainsi que le nombre d'octets
Séquence		0 à 2	L'unité APDU est une demande APDU AND ControlStatus.Instruction indique Chargement segmenté OU Enregistrement segmenté	Indique si cette demande est la première, la dernière, ou si elle est émise au milieu d'un transfert segmenté.

4.2 Types de données

La notation pour les types de données est la même que pour l'IEC 61158-6, Type 1 pour les types suivants:

- Integer, Integer8, Integer16, Integer32
- Unsigned, Unsigned8, Unsigned16
- Floating32, Floating64

5 Syntaxes de transfert

5.1 Encodage des unités APDU

5.1.1 Encodage de l'En-tête des unités APDU

5.1.1.1 Structure de l'en-tête des unités APDU

La syntaxe abstraite de l'en-tête des unités APDU est définie en 4.1.2. Le Paragraphe 5.1 décrit l'encodage de l'en-tête. L'en-tête d'unité APDU est composé de trois champs, comme indiqué à la Figure 2.

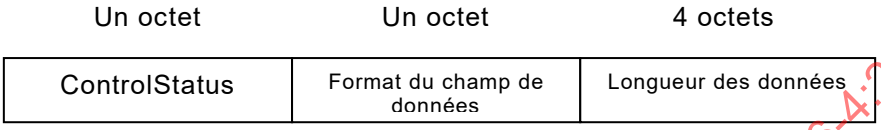


Figure 2 – Structure de l'en-tête des unités APDU

5.1.1.2 ControlStatus

ControlStatus est codé dans un octet. Si l'APDU est une demande APDU, ControlStatus contient les instructions des sous-champs, la méthode d'adressage, SourceID et SecureDataExchange. L'interprétation de cet octet est indiqué à la Figure 3.

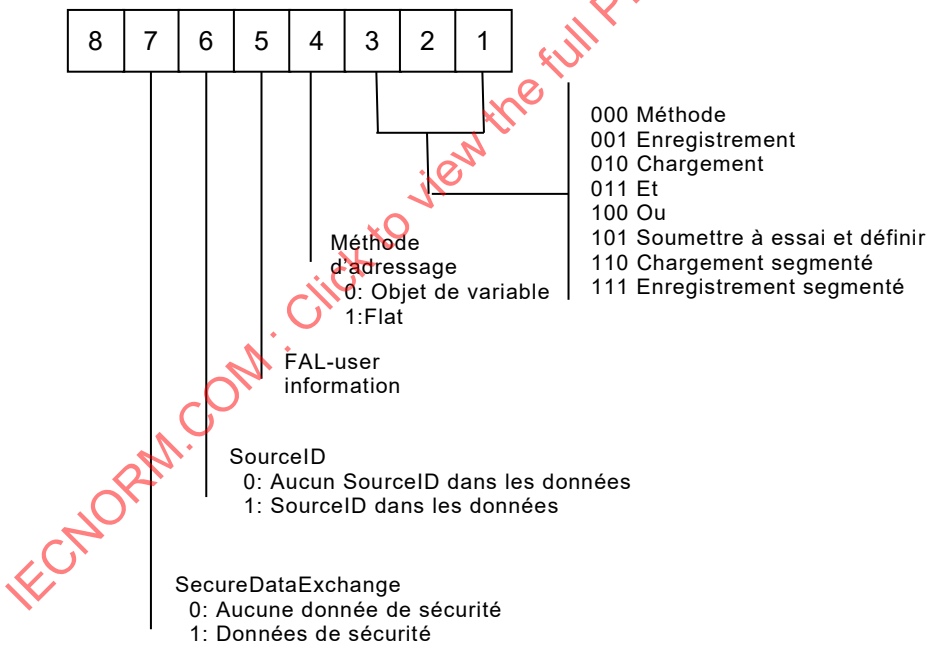


Figure 3 – Sous-champs de ControlStatus pour une demande

Si l'instruction est = 000 (=Errorcode), les cinq bits restants de ControlStatus contiennent le code d'erreur. Les valeurs du code sont indiquées à la Figure 4.

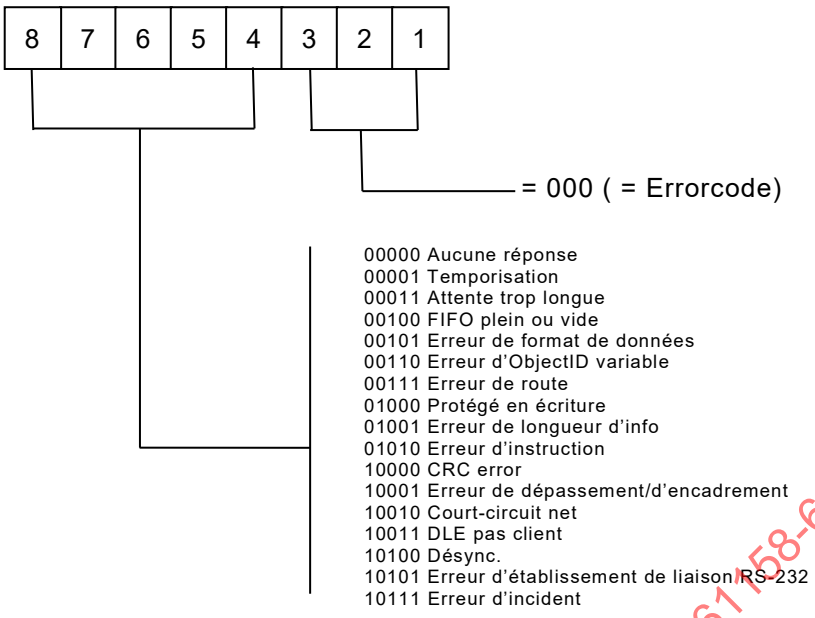


Figure 4 – Sous-champs de ControlStatus pour une réponse avec erreur

Si l'instruction est <> 000 (<> Errorcode), les cinq bits restants de ControlStatus contiennent les sous-champs de Statuscode et HistoricalDataError. Le codage de ces champs est indiqué à la Figure 5.

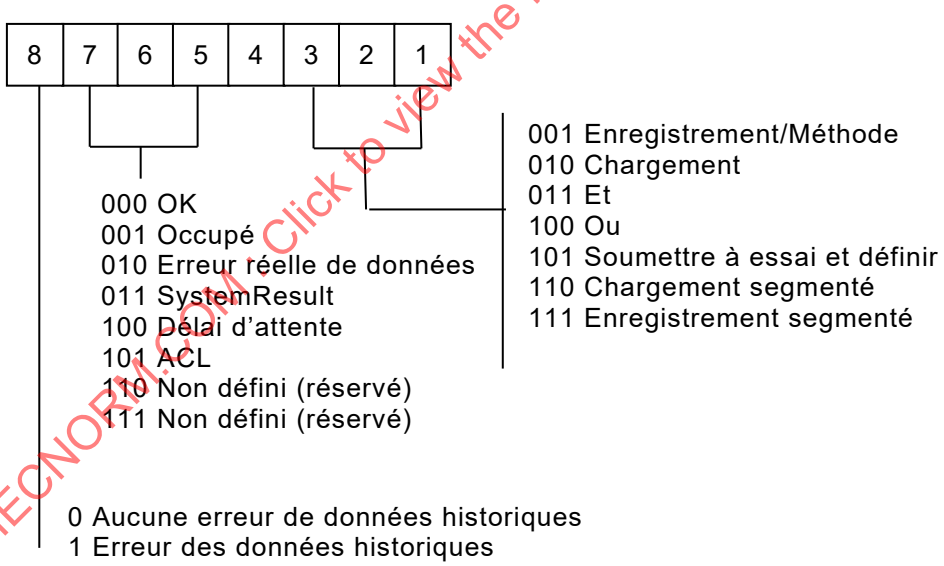


Figure 5 – Sous-champs de ControlStatus pour une réponse sans erreur

5.1.1.3 DataFieldFormat

DataFieldFormat est codé dans un octet. Le codage de cet octet est indiqué à la Figure 6.

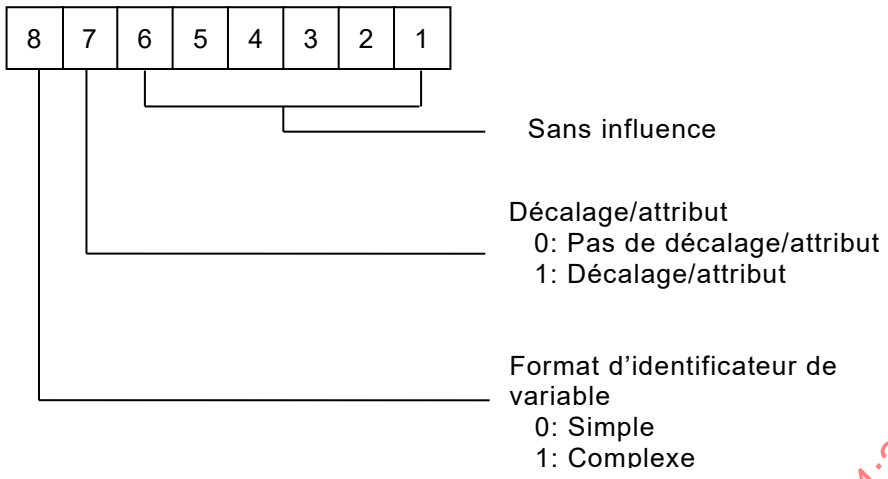


Figure 6 – Codage de DataFieldFormat

5.1.1.4 DataLength

DataLength est un Integer32, indiquant la longueur totale du corps des unités APDU.

5.1.2 Encodage du corps des unités APDU

5.1.2.1 Structure du corps des unités APDU

La syntaxe abstraite du corps des unités APDU est définie en 4.1.3. Le Paragraphe 5.1.2 décrit l'encodage. L'interprétation du corps des unités APDU est indiquée par l'en-tête des unités APDU.

Le corps des unités APDU de demande peut comprendre jusqu'à quatre ou cinq champs possibles, comme indiqué à la Figure 7.

2 ou 4 octets	0, 2 ou 4 octets	0 – Taille PDU max en octets	0 – 2 octets	0 ou 1 octet
Identificateur de variable	Décalage/attribut	Données	Longueur demandée	Séquence

Figure 7 – Structure du corps des unités APDU de demande

Le corps d'une unité APDU de demande peut comprendre jusqu'à deux champs, comme indiqué à la Figure 8.

0, 2 ou 4 octets	0 – Taille PDU max en octets
Décalage/attribut	Données

Figure 8 – Structure du corps des unités APDU de réponse

5.1.2.2 Identificateur de variable

L'identificateur de variable peut être simple ou complexe. S'il est simple, il n'est composé que d'une sous-trame, ID, de type Integer16. S'il est complexe, il est composé de deux sous-trames, code (1 octet) et ID (3 octets), comme indiqué à la Figure 9.

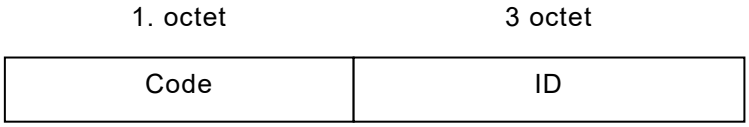


Figure 9 – Identificateur de variable

Le codage de la sous-trame Code de l'identificateur de variable est indiqué à la Figure 10.

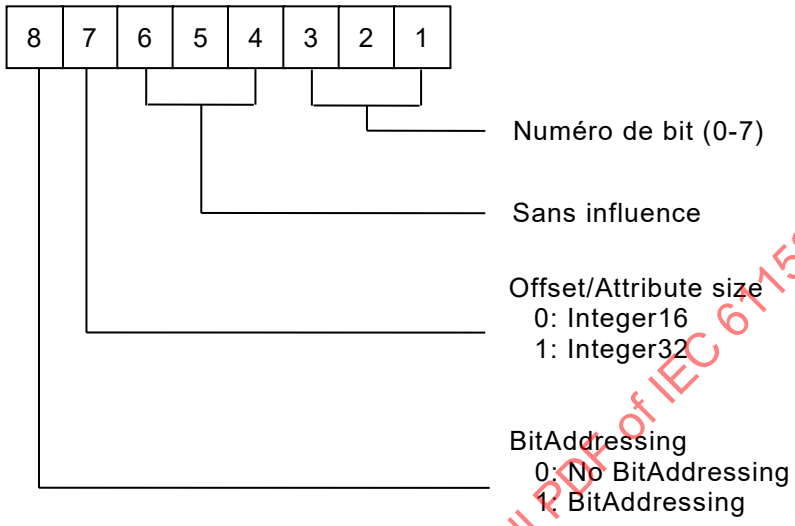


Figure 10 – Sous-trame Code de l'identificateur de variable

5.1.2.3 Décalage/attribut

La sous-trame Offset/Attribute du corps d'unité APDU peut être ou non présente. Dans l'affirmative, Offset/Attribute est un Integer16 ou un Integer32. Une valeur négative sélectionne un attribut de l'objet de variable. Une valeur positive, pour sélectionner une partie de la valeur des données de l'objet de variable, indique le décalage, en octets, par rapport à l'octet de démarrage du bloc de données à transférer, relativement au premier octet de la variable.

5.1.2.4 RequestedLength

La sous-trame RequestedLength peut être ou non présente.

Si l'unité APDU est une unité APDU de demande et que la sous-trame ControlStatus.Instruction de l'en-tête d'unité APDU indique Read ou Segmented Read, alors la sous-trame RequestedLength est présente. Elle indique la longueur en octets des données ou de l'attribut demandés.

La sous-trame RequestedLength est de type Unsigned8 ou Unsigned16. Etant donné qu'il n'existe pas de sous-trame Data dans l'unité APDU s'il existe une sous-trame RequestedLength, la taille de RequestedLength est donnée implicitement par le paramètre DataLength. Si la valeur de RequestedLength est inférieure à 256, RequestedLength doit être de type Unsigned8.

5.1.2.5 Données

La sous-trame Données du corps d'unité APDU peut contenir:

a) Des données codées et condensées comme indiqué en 5.2, ou

b) l'attribut d'un objet de variable, codé et condensé comme indiqué en 5.2.

Etant donné qu'il n'existe pas de sous-trame RequestedLength dans l'unité APDU s'il existe une sous-trame Data, la taille de la sous-trame Data est donnée implicitement par le paramètre DataLength.

5.1.2.6 Séquence

La sous-trame Séquence est d'un octet et présente les valeurs conformes suivantes:

0: Indique qu'il s'agit de la première demande d'un chargement ou enregistrement segmenté.

1: Indique qu'il s'agit de l'une des demandes suivantes d'un chargement ou enregistrement segmenté.

2: Indique qu'il s'agit de la dernière demande d'un chargement ou enregistrement segmenté.

5.2 Encodage et compression des objets de variable

5.2.1 Encodage de variables simples

5.2.1.1 Encodage d'une valeur booléenne

a) L'encodage d'une valeur booléenne doit être de type primitif. La chaîne ContentsOctets doit être formée d'un seul octet.

b) Si la valeur booléenne est FALSE, le bit 1 de la valeur de la chaîne ContentsOctets doit être 0 (zéro). Si la valeur booléenne est TRUE, le bit 1 de la valeur de la chaîne ContentsOctets doit être 1 (un).

5.2.1.2 Encodage d'une valeur entière

Comme défini dans la syntaxe de transfert 1 de l'IEC 61158-6:2003, Type 1, encodage d'une Valeur Integer.

5.2.1.3 Encodage d'une valeur Unsigned

Comme défini dans la syntaxe de transfert 1 de l'IEC 61158-6:2003, Type 1, encodage d'une Valeur Unsigned de types Unsigned8 et Unsigned16.

5.2.1.4 Encodage d'une valeur en virgule flottante

Comme défini dans la syntaxe de transfert 1 de l'IEC 61158-6:2003, Type 1, encodage d'une Valeur Floating-Point.

5.2.2 Encodage de variables construites

5.2.2.1 Encodage d'une valeur String

a) L'encodage d'une valeur eString à longueur variable doit être de type primitif.

b) Le champ Length doit indiquer le nombre d'éléments dans la valeur String par un nombre binaire.

c) Le champ Length doit être dans le premier octet.

5.2.2.2 Encodage d'une valeur BitString

a) L'encodage d'une valeur BitString doit être de type primitif.

b) Il n'existe pas de champ Length dans la valeur BitString.

c) La valeur de BitString, qui commence par le premier bit et va jusqu'au dernier bit, doit être placée sur les bits 1 à 8 du premier octet, puis des bits 1 à 8 du deuxième octet, puis des bits 1 à 8 de chaque octet jusqu'au dernier octet de la chaîne ContentsOctets.

d) Les bits non utilisés, s'il y en a, doivent être placés sur les bits 2 à 8 du dernier octet.

5.2.2.3 Codage d'un résultat de méthode

a) L'encodage d'un résultat de Méthode doit être de type primitif.

b) Le résultat de Méthode peut être une variable simple ou une variable construite.

c) Si le sous-champ Statuscode dans ControlStatus est SystemResult, la longueur des données d'APDU est 2 fois plus élevée que la longueur du résultat de Méthode et les 2 premiers octets des données d'APDU contiennent une information de résultat définie par l'utilisateur relative à la méthode activée dans l'identificateur de variable.

5.2.2.4 Structure du sous-champ du corps d'unité APDU

Les données dans le sous-champ du corps d'unité APDU sont empaquetées dans une séquence comme indiqué à la Figure 11:

Données utilisateur	MethodID	SourceID	Nonce	Signature

Figure 11 – Séquence de données dans le sous-champ du corps d'unité APDU

MethodID est de 2 ou 4 octets en fonction de la valeur de la méthode d'adressage relative à l'instruction Méthode.

SourceID est de 2 octets et n'est présent que dans une APDU de demande lorsque le sous-champ SourceID est SourceIDInData dans Controlstatus.

Nonce est un BitString[64], 8 octets, et n'est présent que dans une APDU de demande lorsque le sous-champ SecureDataExchange est SecureData dans Controlstatus et que l'instruction est Load (Chargement) ou Segmented Load (Chargement segmenté). Si Nonce est présent dans le corps d'unité APDU, Signature n'est pas présent.

Signature est une chaîne de bits[64], 8 octets, et n'est présent que dans l'APDU de demande lorsque le sous-champ SecureDataExchange est SecureData dans ControlStatus. Dans le cas d'une APDU de demande, Signature est uniquement présent si l'instruction est Method (Méthode), Store (Enregistrement), And (Et), Or (Ou) ou Segmented Store (Enregistrement segmenté). Dans le cas d'une APDU de réponse, Signature est présent si SecureData a fait l'objet d'une demande dans SecureDataExchange. Si Signature est présent dans le corps de l'unité APDU, Nonce n'est pas présent.

5.2.2.5 Calcul de Signature

Signature (code d'authentification), type de données BitString[64], est généré comme suit:

HFUNC(Key, Param, MSG)

Où HFUNC est une fonction de hachage cryptographique utilisant une clé Key de bits Key, type de données Bitstring[128], qui est commune à la fois au demandeur et au répondeur. Pour une demande de type Read (Lecture), Param est un MSG provenant de la demande reçue. Pour une demande de type Write (Ecriture) ou Method, Param est le Nonce qui a été précédemment reçu par le Serveur (également présenté dans l'exemple en 8.2.2.3.1). Pour une réponse de type Write ou Method, Param est le Nonce qui a été précédemment reçu par le demandeur (également présenté dans l'exemple en 8.2.2.3.1). Nonce est un type de données BitString[64]. MSG est défini comme indiqué à la Figure 12.

HFUNC, tel que défini dans l'algorithme 1 MAC de l'ISO/IEC 9797-1 (AES-CBC-MAC).

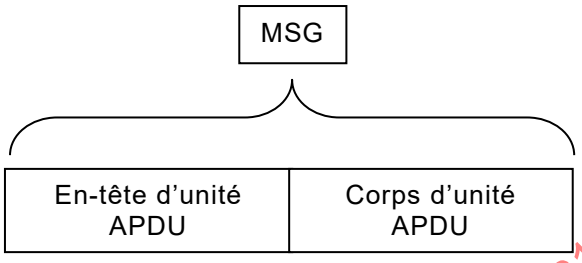


Figure 12 – MSG se compose de l'en-tête d'unité APDU et du corps d'unité APDU

5.2.3 Alignement

5.2.3.1 Généralités

Le Paragraphe 5.2.2.3 explique comment les champs et les éléments des variables construites sont alignés et transférés.

L'alignement général est de deux.

- Les champs et les éléments de type de base dont la taille est de 1 octet doivent être transférés immédiatement après le champ ou l'élément précédent.
- Les champs et éléments d'une taille supérieure à 1 octet doivent être transférés suivant un décalage identique par rapport au premier octet de la variable construite.
- Les champs et éléments de type construit doivent être transférés suivant un décalage identique par rapport au premier octet de la variable construite.

5.2.3.2 Syntaxe de transfert des éléments de matrice

Le Tableau 4 présente un exemple d'une syntaxe de transfert d'une variable "ArrayVar" de type

ARRAY[1..2] of ARRAY[1..3] of Integer8.

Tableau 4 – Syntaxe de transfert des matrices

1. octet	ArrayVar[1,1]
2. octet	ArrayVar[1,2]
3. octet	ArrayVar[1,3]
4. octet	ArrayVar[2,1]
5. octet	ArrayVar[2,2]
6. octet	ArrayVar[2,3]

5.2.3.3 Syntaxe de transfert des champs de structure

Le Tableau 5 présente un exemple d'une syntaxe de transfert d'une variable "StructVar" de type

```
STRUCTURE
      Field1: Integer8;
      Field2: STRUCTURE
                Sub1: Integer16;
                Sub2: BitString[8];
         END;
      Field3: Integer8;
      Field4: BitString[8];
      Field5: Integer8;
END;
```

Tableau 5 – Syntaxe de transfert de structure

1. octet	StructVar.Field1
2. octet	Dummy
3. octet	StructVar.Field2.Sub1, octet le plus important
4. octet	StructVar.Field2.Sub1, octet le moins important
5. octet	StructVar.Field2.Sub2
6. octet	StructVar.Field3
7. octet	StructVar.Field4
8. octet	StructVar.Field5

5.2.4 Attributs d'objets de variable

Pour tous les objets de variable, les attributs facultatifs peuvent être définis dans le Tableau 6.

Tableau 6 – Attributs communs des objets de variable

Nom d'attribut	Index	Format
Identificateur du type de variable	-20	Unsigned8
Longueur en octets	-24	Integer32
Activation en lecture	-28	Booléen
Activation en écriture	-29	Booléen
Protégé en écriture	-30	Booléen

L'attribut clé, l'identificateur de variable, permet d'identifier l'objet de variable, et n'est pas un attribut accessible.

Les attributs d'un objet de variable peuvent être accessibles à partir du réseau. Si un objet de variable ne permet pas d'accéder à un attribut, il doit répondre à une tentative d'accès avec le code d'erreur "Variable Object ID error". On ne peut accéder qu'à un seul attribut puisqu'une seule invocation de services n'a été effectuée.

Pour chaque objet de variable, il n'existe qu'un seul ensemble d'attributs. Cela signifie que les sous-trames ou les éléments des variables construites n'ont pas leurs propres attributs.

Le Tableau 7 affiche les valeurs de l'identificateur de variable pour les différents types de variables.

Tableau 7 – Identificateurs des différents types de variables

Type de variable	Identificateur
Booléen	43
Integer8	49
Integer16	34
Integer32	35
Unsigned8	48
Unsigned16	50
Float32	36
Float64	37
UNICODE Char	51
Complexe	61
String	40
BitString	62
FIFO	63

Les objets de variable de type FIFO doivent présenter les attributs supplémentaires obligatoires définis dans le Tableau 8.

Tableau 8 – Attributs d'objets de variable de type FIFO

Nom d'attribut	Index	Format
Elément d'entrée suivant	-2	Integer16
Elément de sortie suivant	-4	Integer16
Eléments libres	-6	Integer16
Eléments utilisés	-8	Integer16
Relire	-9	Booléen
Réécrire	-10	Booléen

5.3 Codes d'erreur

Le Paragraphe 5.3 spécifie les valeurs hexadécimales pour les codes d'erreur dans le paramètre d'état d'erreur de la variable ASE service RESPONSE. Les valeurs du code sont indiquées dans le Tableau 9.

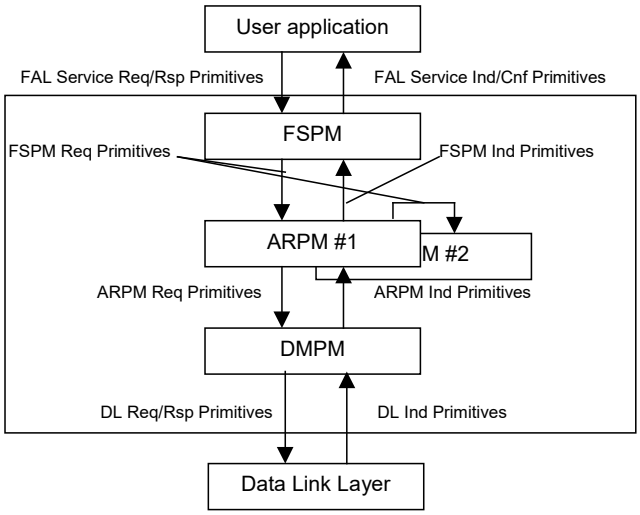
Tableau 9 – Codes d'erreur

Description de l'erreur	Code d'erreur (binaire)
Erreur d'instruction	01010000
Erreur de longueur d'info	01001000
FIFO plein ou vide	00100000
Protégé en écriture	01000000
Erreur de format de données	00101000
Erreur d'ID d'objet variable	00110000
Temporisation	00001000
Erreur des données réelles	x010xaaa
Erreur des données historiques	1xxxxaaa
Erreur de route	00111000
Pas de réponse	00000000
Délai d'attente trop long	00011000
Désync.	10100000
CRC error	10000000
DLE pas client	10011000
Court-circuit net	10010000
erreur de dépassement/d'encadrement	10001000
Erreur d'établissement d'une liaison RS-232	10101000
Erreur de signature	10111000
"x" signifie que le bit n'a pas d'importance. "aaa" signifie qu'au moins un des trois bits est vrai (1).	

6 Diagrammes d'états de protocole de la couche FAL

Les alinéas suivants décrivent l'interface avec les services et les machines de protocoles de couche FAL.

Le comportement de la couche FAL est décrit par trois machines de protocole intégrées. Les trois machines de protocole sont les suivantes: machine de protocole de service FAL (FAL Service Protocol Machine, FSPM), machine de protocole de relations entre applications (Application Relationship Protocol, ARPM), machine de protocole de mapping de couche Liaison de données (Data-link Layer Mapping Protocol Machine, DMPM). Les relations et les primitives échangées entre ces machines de protocole sont décrites dans la Figure 13.



Anglais	Français
User application	Application utilisateur
FAL Service Req/Rsp Primitives	Primitives Req/Rsp de service FAL
FAL Service Ind/Cnf Primitives	Primitives Ind/Cnf de service FAL
FSPM Req Primitives	Primitives FSPM Req
FSPM Ind Primitives	Primitives FSPM Ind
ARPM Req Primitives	Primitives ARPM Req
DL Req/Rsp Primitives	Primitives de Req/Rsp DL
DL Ind Primitives	Primitives DL Ind
Data Link Layer	Couche de liaison de données
ARPM #1	ARPM n° 1
ARPM #2	ARPM n° 2

Figure 13 – Résumé de l'architecture FAL

La machine FSPM décrit l'interface de service entre l'utilisateur FAL et un REP. La machine de protocole FSPM est chargée des activités suivantes.

– Accepter les primitives de service émises par l'utilisateur de service FAL et les convertir en primitives internes FAL.
– Sélectionner le point AREP identifié par le paramètre du chemin de destination fourni par l'application de l'utilisateur et envoyer les primitives FAL internes au point AREP choisi.
– Accepter les primitives FAL internes provenant du point AREP, exécuter les actions indiquées dans ces primitives et renvoyer le résultat au point AREP, d'où elles sont reçues. Cette action s'applique aux indications présentant des unités APDU de demande.
– Accepter les primitives FAL internes provenant du point AREP, les convertir en service et envoyer ces primitives de service à l'utilisateur FAL, à la suite de l'invocation de l'utilisateur FAL du service RESPONSE. Cette action s'applique aux indications présentant des unités APDU de réponse.

La machine ARPM décrit l'échange d'unités de données de protocole (Protocol Data Unit, PDU) de couche FAL avec des machines ARPM distantes. Elle ne subit aucun changement d'état. La machine de protocole ARPM est chargée des activités suivantes:

– Accepter les primitives FAL internes provenant de la machine FSPM et créer et envoyer d'autres primitives FAL internes à la machine DMPM.

– Accepter les primitives FAL internes de la machine DMPM et les envoyer à la machine FSPM en tant qu'indications, ou à une autre machine ARPM en tant que demandes.

La machine DMPM décrit le mapping entre la couche FAL et la couche DLL. Elle ne subit aucun changement d'état. La machine DMPM est chargée des activités suivantes.

– Accepter les primitives FAL internes provenant de la machine ARPM, préparer les primitives de service DLL et
– Recevoir les primitives d'indication provenant de la couche DLL et les envoyer à la machine ARPM sous forme de primitives FAL internes.

7 Diagramme d'états de contexte AP

Le diagramme d'états de contexte AP n'est pas présent.

8 Machine de protocole de service FAL (FSPM)

8.1 Primitives échangées entre l'utilisateur FAL et la machine FSPM

Les primitives échangées entre l'utilisateur FAL et la machine FSPM sont indiquées dans le Tableau 10.

Tableau 10 – Primitives échangées entre l'utilisateur FAL et la machine FSPM

Nom de la primitive	Source	Paramètres associés	Commentaires
REQUEST.req	Utilisateur FAL	REP, Variable Object ID, Variable Service, Data length, Offset/Attribute, Bit-no, Data	Cette primitive permet de transmettre une demande d'utilisateur FAL à un REP
RESPONSE.cnf	FSPM	REP, Variable Service, Data length, Error status, Data	Cette primitive permet de transmettre une réponse d'un REP à l'utilisateur FAL

8.2 Etats FSPM

8.2.1 Généralités

Les états suivants sont définis pour un REP: IDLE RESERVED, WAITING FOR RESPONSE, RESPONSE RECEIVED, NOT IN USE.

Un REP peut jouer le rôle d'un objet proxy (client) ou d'un objet réel (serveur). Etant donné que les REP des objets proxy et les REP des objets réels ont des comportements très différents, ils sont décrits dans des paragraphes différents.

8.2.2 Etats des objets proxy de la machine FSPM

8.2.2.1 Présentation

Les états suivants s'appliquent à un REP dans le rôle de l'objet proxy:

NON UTILISE

Le REP n'est pas en cours d'utilisation. La seule primitive de service autorisée est ReserveREP.req. Toutes les autres primitives doivent être rejetées.