# IEC 61158-6-4

Edition 4.0 2023-03

# INTERNATIONAL STANDARD

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



IEC 61158-6-4:2023-03(en)

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

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

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

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

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 It also gives information on projects, replaced and withdrawn publications.

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

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

**IEC Products & Services Portal - products.iec.ch**
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

**Electropedia - www.electropedia.org**
The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

# IEC 61158-6-4

Edition 4.0 2023-03

# INTERNATIONAL STANDARD

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



INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-6632-8

**Warning! Make sure that you obtained this publication from an authorized distributor.**

CONTENTS

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 the IEC 61784-1 series and the IEC 61784-2 series.

IEC 61158-6-4 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

This fourth edition cancels and replaces the third edition published in 2019. This edition constitutes a technical revision.

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

a) Use of extended data size in an APDU body. This extension is restricted to nodes operating on a P-NET IP network.

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

| Draft | Report on voting |
|---|---|
| 65C/1204/FDIS | 65C/1245/RVD |

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

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2 and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

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 document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document 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 can 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 document specifies interactions between remote applications and defines the externally visible behavior provided by the Type 4 fieldbus application layer in terms of

- the formal abstract syntax defining the application layer protocol data units conveyed between communicating application entities;
- the transfer syntax defining encoding rules that are applied to the application layer protocol data units;
- the application context state machine defining the application service behavior visible between communicating application entities;
- 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

- define the wire-representation of the service primitives defined in IEC 61158-5-4, and
- 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 the IEC 61784-1 series and the IEC 61784-2 series 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 61158-3-4:2023, *Industrial communication networks – Fieldbus specifications – Part 3-4: Data-link layer service definition – Type 4 elements*

IEC 61158-4-4:2023, *Industrial communication networks – Fieldbus specifications – Part 4-4: Data-link layer protocol specification – Type 4 elements*

IEC 61158-5-4:2023, *Industrial communication networks – Fieldbus specifications – Part 5-4: Application layer service definition – Type 4 elements*

IEC 61158-6-1, *Industrial communication networks – Fieldbus specifications – Part 6-1: Application layer protocol specification – Type 1 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) – Part 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, abbreviated terms and conventions

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

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

- IEC Electropedia: available at https://www.electropedia.org/
- ISO Online browsing platform: available at https://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<br>=><br>action | Next state |
|---|---|---|---|
| Name of this transition | The current state to which this state transition applies | Events or conditions that trigger this state transaction.<br>=><br>The actions that are taken when the above events or<br>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 can 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<br>Method<br>Store<br>Load<br>And<br>Or<br>Test-And-Set<br>Segmented Load<br>Segmented Store | | |
| ControlStatus | Errorcode | Described in Figure 3 to Figure 5 | ControlStatus.Instruction = Errorcode | |
| ControlStatus | Addressing method | Variable Object<br>Flat | ControlStatus.Instruction <> Errorcode | |

| Field name | Subfield name | Possible values | Constraint (present if) | Comment |
|---|---|---|---|---|
| ControlStatus | Addressing method | Variable Object<br>Flat | ControlStatus.Instruction =Method | Variable object means 2 octet MethodID<br>Flat means 4 octet MethodID |
| ControlStatus | SourceID | NoSourceIDInData<br>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<br>SecureData | APDU is a request APDU | Read type instruction: Used by the requesting application to ensure an authenticated response.<br>Write type instruction: Used by the requesting application to ensure an authenticated write type instruction in the responder. |
| ControlStatus | ActualDataError | NoActualError<br>ActualError | ControlStatus.Instruction <> Errorcode | Used by the responding user application to indicate, that an actual error can affect the accessed Variable Object |
| ControlStatus | SystemResult | NoSystemResult<br>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<br>Hence, DataLength is 2 higher than the method result length. |
| ControlStatus | HistoricalDataError | NoHistoricalError<br>HistoricalError | ControlStatus.Instruction <> Errorcode | Used by the responding user application to indicate that an error can have affected the accessed Variable Object |
| DataFieldFormat | Offset/Attribute | No Offset/Attribute<br>Offset/Attribute | | Indicates, whether the APDU Body holds an Offset/Attribute field |
| DataFieldFormat | Variable Identifier Format | Simple<br>Complex | APDU is a request APDU | Indicates the format of the Variable Identifier in a request APDU |
| DataFieldFormat | Offset/Attribute size | Integer16<br>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<br>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<br>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<br>Integer32 | APDU is a request APDU AND DataFieldFormat.VariableIdentifierFormat = Complex AND<br>DataFieldFormat.Offset/Attribute = Offset/Attribute | |
| VariableIdentifier | ID | −32 768 to<br>+32 767 | APDU is a request APDU AND DataFieldFormat.VariableIdentifierFormat = Simple | |
| VariableIdentifier | ID | −8 388 608 to<br>+8 388 607 | APDU is a request APDU AND DataFieldFormat.VariableIdentifierFormat = Complex | |
| Offset/Attribute | | −32 768<br>+32 767 | APDU is a request APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND<br>VariableIdentifier.Code.Offset/Attribute size = Integer16 | Negative values select attribute, positive values select part of constructed variable |
| Offset/Attribute | | −2 147 483 648 to<br>+2 147 483 647 | APDU is a request APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND<br>VariableIdentifier.Code.Offset/Attribute size = Integer32 | Negative values select attribute, positive values select part of constructed variable |
| Offset/Attribute | | −32 768 to<br>+32 767 | APDU is a response APDU AND DataFieldFormat.Offset/Attribute = Offset/Attribute AND<br>DataFieldFormat.Offset/Attribute size= Integer16 | Negative values select attribute, positive values select part of constructed variable |

| Field name | Subfield name | Possible values | Constraint (present if) | Comment |
|---|---|---|---|---|
| Offset/Attribute | | −2 147 483 648 to +2 147 483 647 | APDU is a response APDU AND DataFieldFormat.Offset/ Attribute = Offset/Attribute AND<br>DataFieldFormat.Offset/ Attribute size= Integer32 | Negative values select attribute, positive values select part of constructed variable |
| Data | | Any | | |
| 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 61158-6-1 Type 1 elements, 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.

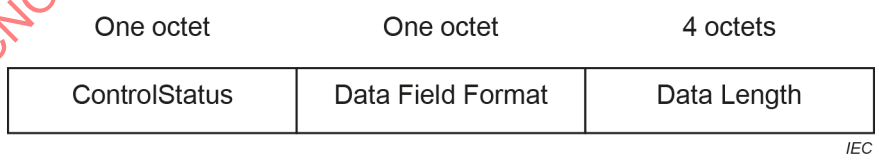


**Figure 2 – APDU header structure**

##### 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.

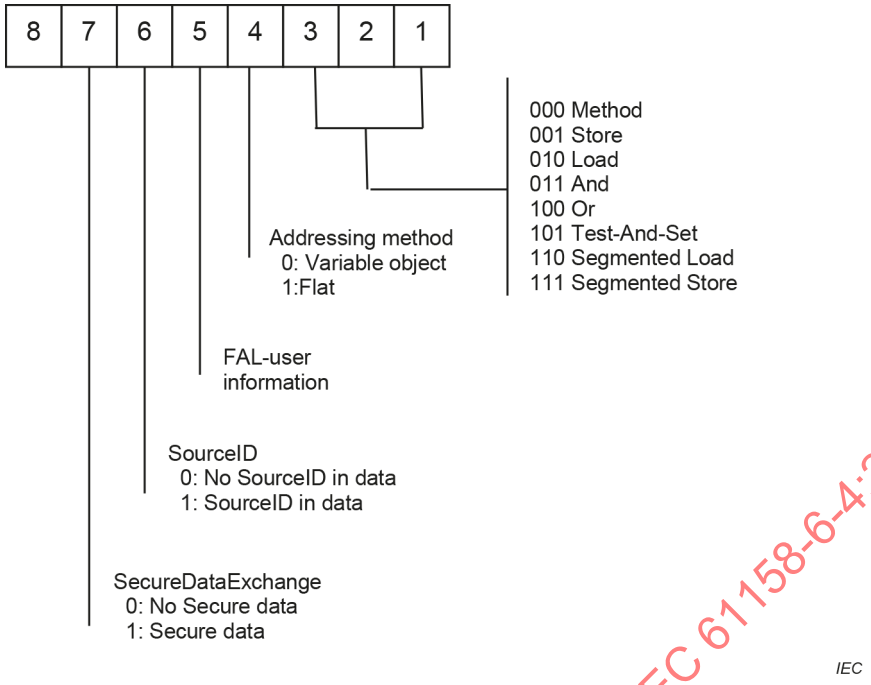


**Figure 3 – Subfields of ControlStatus for Request**

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

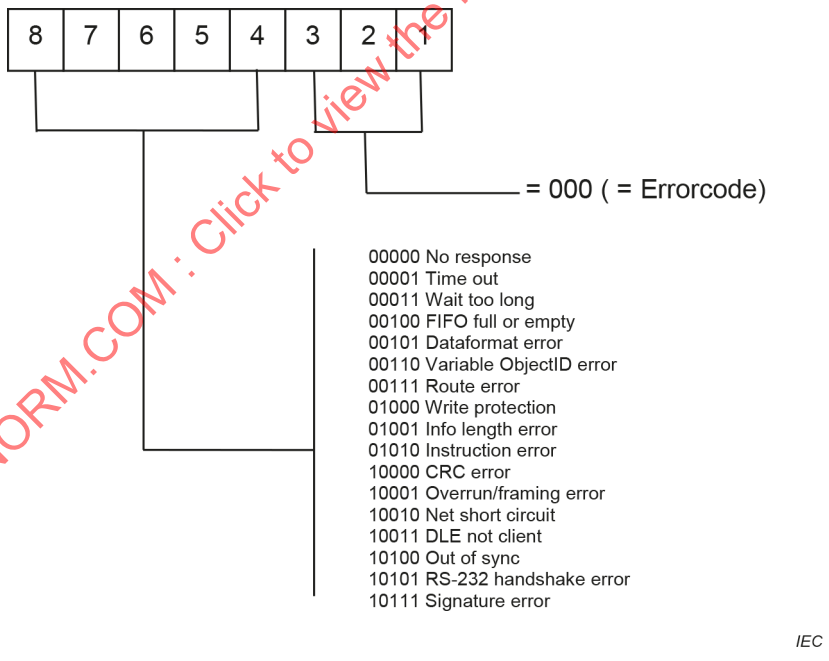


**Figure 4 – Subfields of ControlStatus for Response with error**

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.

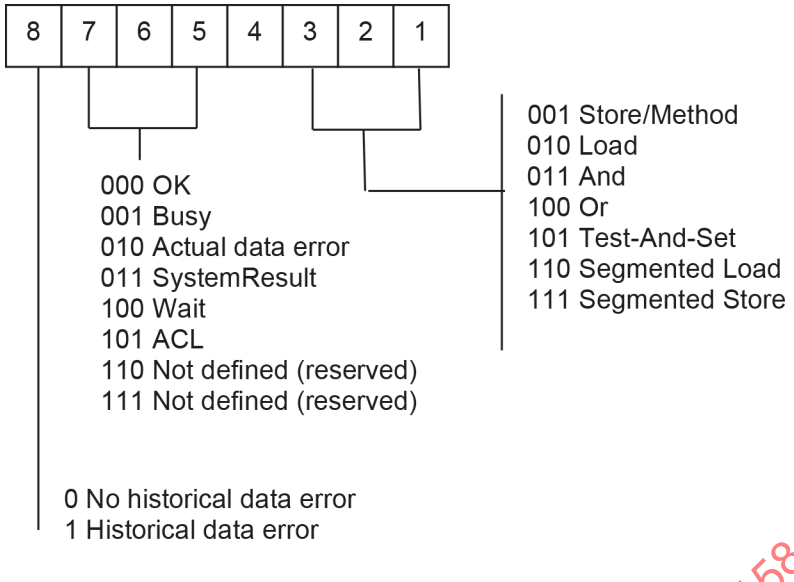


**Figure 5 – Subfields of ControlStatus for Response with no error**

#### 5.1.1.3 DataFieldFormat

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

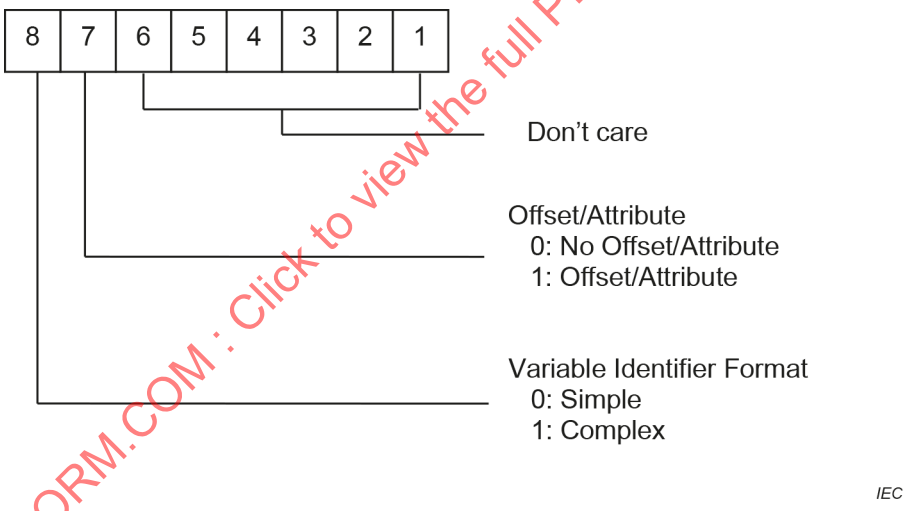


**Figure 6 – DataFieldFormat encoding**

#### 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 can consist of up to four of five possible fields, as shown in Figure 7.

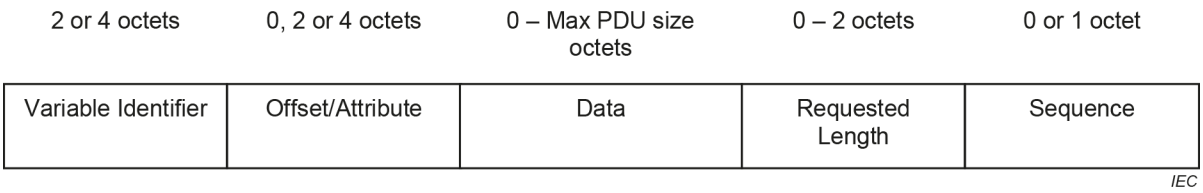


**Figure 7 – Structure of request APDU body**

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

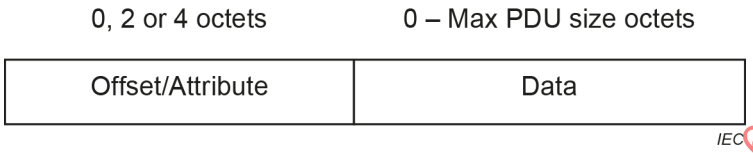


**Figure 8 – Structure of response APDU body**

### 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.

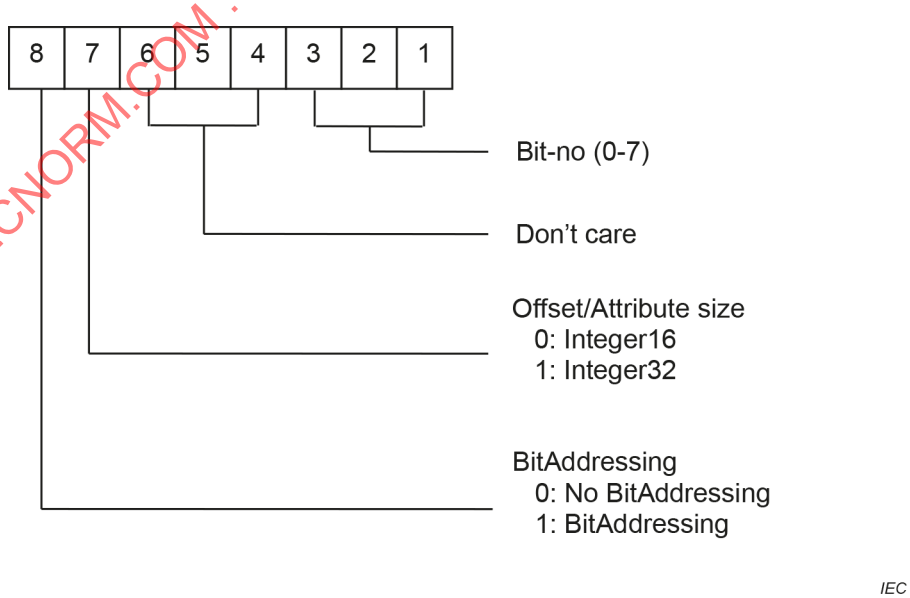


**Figure 10 – Code subfield of variable identifier**

#### 5.1.2.3 Offset/attribute

It is possible that the Offset/Attribute subfield of the APDU body is present, or not. 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

It is possible that the RequestedLength subfield is present, or not.

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 61158-6-1 Type 1 elements, Transfer syntax 1, Encoding of an Integer Value.

##### 5.2.1.3 Encoding of an Unsigned value

As defined in IEC 61158-6-1 Type 1 elements, 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 61158-6-1 Type 1 elements, 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:

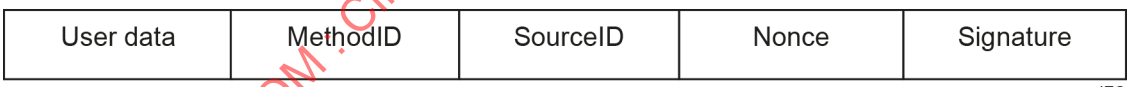


**Figure 11 – Sequence of data in the APDU body subfield**

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).

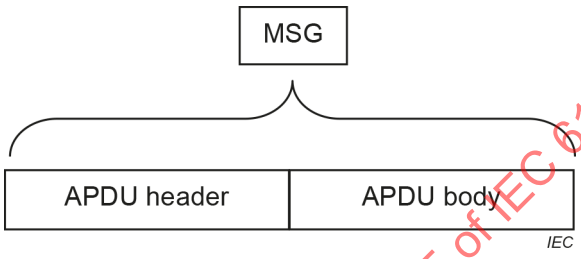


**Figure 12 – MSG consists of APDU header and APDU body**

### 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**

| Octet number | Array element |
|---|---|
| 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**

| Octet number | Structure element |
|---|---|
| 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 can 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 can 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.<br>"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.

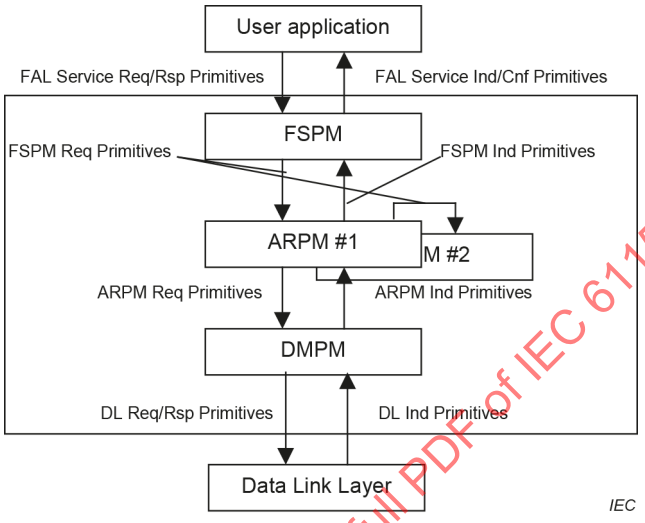


**Figure 13 – Summary of FAL architecture**

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.

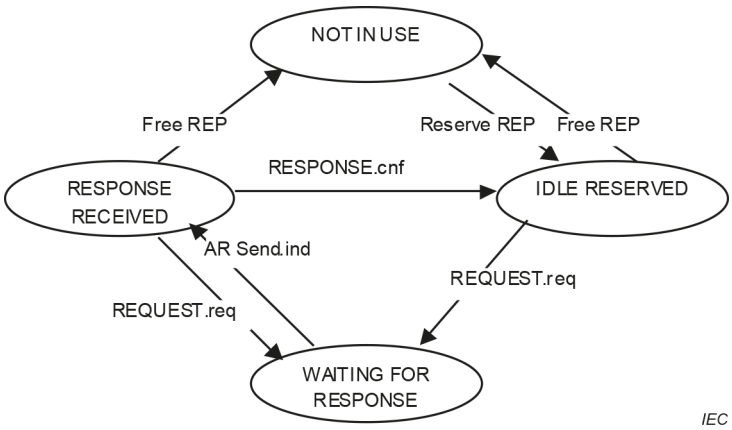


**Figure 14 – FSPM proxy object state machine**

### 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 can 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 can 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.<br>If the Route-info.Destination-Route defines an IP Type 4 route format and includes a gateway element and addressed AREP attribute MaxDataSize is greater than 56, then addressed AREP attribute MaxDataSize is set to 56 for this Request |
| 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:<br>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<br>If Variable Service parameter is Write and parameter Data.Bit1 is TRUE then<br>Set APDU Header.ControlStatus.Instruction to Or, and<br>clear all bits in parameter Data and set bit indicated by Local Bit-no<br>If Variable Service parameter is Write and parameter Data.Bit1 is FALSE then<br>Set APDU Header.ControlStatus.Instruction to And, and<br>Set all bits in parameter Data and clear bit indicated by Local Bit-no<br>If Variable Service parameter is And, Or or Test-And-Set then<br>Set APDU Header.ControlStatus.Instruction to And, Or, Test-And-Set respectively, and<br>rotate parameter Data.Bit1 to position indicated by Local Bit-no<br>If Variable Service parameter is Read then<br>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:<br>Parameter Variable Service Method -> Store<br>Parameter Variable Service Write -> Store<br>Parameter Variable Service Read -> Load<br>Parameter Variable Service And -> And<br>Parameter Variable Service Or -> Or<br>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 |

| Action ID | Action description |
|---|---|
| 13 | If<br>parameter Bit-no indicates bit-addressing, or<br>the value of parameter Variable Object ID is lower than –32768 or higher than +32767, or<br>the value of parameter Offset/Attribute is lower than –32768 or higher than +32767,<br>then set APDU Header DataFieldFormat.Variable Identifier Format to Complex, and set APDU Header Data length to 4.<br>If this does not apply, then set APDU Header DataFieldFormat.Variable Identifier Format to Simple, and set APDU Header Data length to 2 |
| 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 56, and set APDU Body Data first 2 octets to "addressed AREP attribute MaxDataSize", if that value is higher than 56. 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 "addressed AREP attribute MaxDataSize", if that value is less than or equal to 56, and set APDU Body Data first 2 octets to "addressed AREP attribute MaxDataSize", if that value is higher than 56, and increment APDU Header Data length by 1 or 2, depending on size of RequestedLength |
| 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 RECEIVED, 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.<br>Copy received APDU Header ControlStatus to local variable ControlStatus.<br>Copy accumulated received APDU Header Data length to local variable Data length.<br>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.

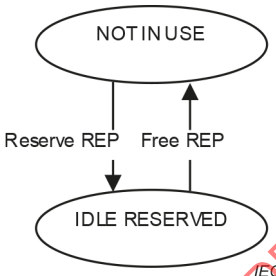


**Figure 15 – FSPM real object state machine**

### 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.<br><br>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.<br><br>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 |