IEC 61158-6-9

Edition 3.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications –
Part 6-9: Application layer protocol specification – Type 9 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-9: Spécification du protocole de la couche application – Eléments de type 9

IEC 61158-6-9:2014-08(en-fr)

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

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

IEC 61158-6-9

Edition 3.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 6-9: Application layer protocol specification – Type 9 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-9: Spécification du protocole de la couche application – Eléments de type 9

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XD

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1760-3

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 10
3.1 Terms and definitions from other ISO/IEC standards 10
3.2 IEC 61158-1 terms 11
3.3 Abbreviations and symbols 14
3.4 Conventions 15
3.5 Conventions used in state machines 16
4 Abstract syntax 17
4.1 FAL-AR PDU abstract syntax 17
4.2 Abstract syntax of PDUBody 19
4.3 Type definitions for ASEs 22
4.4 Abstract syntax of data types 27
5 Transfer syntax 28
6 Structure of FAL protocol state machines 39
7 AP-Context state machines 40
7.1 VCR PM structure 40
7.2 VCR PM state machine 41
8 FAL service protocol machine (FSPM) 53
8.1 General 53
8.2 FSPM state tables 53
8.3 Functions used by FSPM 56
8.4 Parameters of FSPM/ARPM primitives 56
9 Application relationship protocol machines (ARPMs) 56
9.1 AREP mapping to data-link layer 56
9.2 Application relationship protocol machines (ARPMs) 66
9.3 AREP state machine primitive definitions 82
9.4 AREP state machine functions 83
10 DLL mapping protocol machine (DMPM) 84
10.1 DMPM States 84
10.2 DMPM state table 85
10.3 Primitives exchanged between data-link layer and DMPM 91
10.4 Functions used by DMPM 93
Bibliography 95

Figure 1 – Insertion of identification information in the FMS PDU 29
Figure 2 – Identification 30
Figure 3 – Coding with identification 31
Figure 4 – Coding without identification 31
Figure 5 – Representation of the value true 31

Figure 6 – Representation of the value false 31
Figure 7 – Coding of data of data type Integer16 32
Figure 8 – Coding of data of data type Unsigned16 32
Figure 9 – Coding of data of data type Floating Point 33
Figure 10 – Coding of data of data type Visible String 33
Figure 11 – Coding of data of data type Octet String 34
Figure 12 – Coding of data of type Date 34
Figure 13 – Coding of data of data type Time-of-day 35
Figure 14 – Coding of data of data type Time-difference 36
Figure 15 – Coding of data of data type Bit String 36
Figure 16 – Coding of data of data type Time-value 37
Figure 17 – Coding of data of user data definitions with identifier 37
Figure 18 – Coding of data of user data definitions without identifier 37
Figure 19 – Coding of ID info for a SEQUENCE 38
Figure 20 – Relationships among protocol machines and adjacent layers 39
Figure 21 – Relationships among protocol machines and adjacent layers 40
Figure 22 – VCR state machine 41
Figure 23 – State transition diagram of FSPM 53
Figure 24 – State transition diagram of the QUU ARPM 67
Figure 25 – State transition diagram of QUB ARPM 69
Figure 26 – State transition diagram of the BNU ARPM 77
Figure 27 – State transition diagram of DMPM 85

Table 1 – Conventions used for state machines 16
Table 2 – Coding for Date type 34
Table 3 – AP-VCR state machine transactions 42
Table 4 – Primitives issued by FAL-User to VCR PM 51
Table 5 – Primitives issued by VCR PM to FAL-User 51
Table 6 – Primitives issued by VCR PM to FSPM 52
Table 7 – Primitives issued by FSPM to VCR PM 52
Table 8 – FSPM state table – sender transactions 54
Table 9 – FSPM state table – receiver transactions 55
Table 10 – Function SelectArep() 56
Table 11 – Parameters used with primitives exchanged between FSPM and ARPM 56
Table 12 – QUU ARPM states 67
Table 13 – QUU ARPM state table – sender transactions 67
Table 14 – QUU ARPM state table – receiver transactions 68
Table 15 – QUB ARPM states 68
Table 16 – QUB ARPM state table – sender transactions 69
Table 17 – QUB ARPM state table – receiver transactions 71
Table 18 – BNU ARPM states 77
Table 19 – BNU ARPM state table – sender transactions 78
Table 20 – BNU ARPM state table – receiver transactions 79

Table 21 – Primitives issued from ARPM to DMPM 82
Table 22 – Primitives issued by DMPM to ARPM 82
Table 23 – Parameters used with primitives exchanged between ARPM and DMPM 82
Table 24 – Function GetArepId() 83
Table 25 – Function BuildFAS-PDU 84
Table 26 – Function FAS_Pdu_Type 84
Table 27 – Function AbortIdentifier 84
Table 28 – Function AbortReason 84
Table 29 – Function AbortDetail 84
Table 30 – DMPM state descriptions 85
Table 31 – DMPM state table – sender transactions 85
Table 32 – DMPM state table – receiver transactions 88
Table 33 – Primitives exchanged between data-link layer and DMPM 91
Table 34 – Function PickArep 93
Table 35 – Function FindAREP 93
Table 36 – Function LocateQubArep 94
Table 37 – Function SetIdentifier() 94

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-9: Application layer protocol specification – Type 9 elements

FOREWORD

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

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

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

International Standard IEC 61158-6-9 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 2010. This edition constitutes a technical revision. The main change with respect to the previous edition is listed below:

- Correct Time-difference valid range
- Correct Table 3 state transition
- Include Transparent timeliness class in BNU AREP formal model

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

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

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

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

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

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

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

INTRODUCTION

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

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-9: Application layer protocol specification – Type 9 elements

1 Scope

1.1 General

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

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

This standard defines in an abstract way the externally visible behavior provided by the Type 9 fieldbus Application Layer in terms of

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

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

1) define the wire-representation of the service primitives defined in IEC 61158-5-9, and
2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 9 IEC fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI Application Layer Structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-9.

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.

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems. Conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

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

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

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

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

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

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

IEC 61158-5-9, *Industrial communication networks – Fieldbus specifications – Part 5-9: Application layer service definition – Type 9 elements*

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

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

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

ISO/IEC 8825-1, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

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

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

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

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 Terms and definitions from ISO/IEC 7498-1

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

3.1.2 Terms and definitions from ISO/IEC 9545

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

3.1.3 Terms and definitions from ISO/IEC 8824-1

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

3.1.4 Terms and definitions from ISO/IEC 8825-1

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

3.2 IEC 61158-1 terms

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.12
behaviour
indication of how the object responds to particular events

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

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

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

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

3.2.15
class code
unique identifier assigned to each object class

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

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

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

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

3.2.18
conveyance path
unidirectional flow of APDUs across an application relationship

3.2.19
cyclic
events which repeat in a regular and repetitive manner

3.2.20
dedicated AR
AR used directly by the FAL User

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

3.2.21
device
physical hardware connection to the link

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

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

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

3.2.24
endpoint
one of the communicating entities involved in a connection

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

3.2.26
error class
general grouping for error definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.3 Abbreviations and symbols

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

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

3.4 Conventions

3.4.1 General concept

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of three parts: its class definitions, its services, and its protocol specification. The first two are contained in IEC 61158-5 subseries. The protocol specification for each of the ASEs is defined in this standard.

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

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

3.4.2 Conventions for class definitions

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

3.4.3 Abstract syntax conventions

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

3.5 Conventions used in state machines

The state machines are described in Table 1:

Table 1 – Conventions used for state machines

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

The conventions used in the state machines are as follows:

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

xxx A parameter name.

Example:
Identifier := reason
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"

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

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

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

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

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

4 Abstract syntax

4.1 FAL-AR PDU abstract syntax

4.1.1 Top level definition

The productions defined here shall be used with the rules for APDU encoding (see 5.1.2).

```
APDU ::= CHOICE {
            [PRIVATE 0] ConfirmedSend-RequestPDU,
            [PRIVATE 1] ConfirmedSend-ResponsePDU,
            [PRIVATE 2] UnconfirmedSend-PDU,
            [PRIVATE 3] UnconfirmedAcknowledgedSend-CommandPDU,
            [PRIVATE 4] Establish-RequestPDU,
            [PRIVATE 5] Establish-ResponsePDU,
            [PRIVATE 6] Establish-ErrorPDU,
            [PRIVATE 7] Abort-PDU,
            [PRIVATE 8] DataSendAcknowledge-PDU
}
```

4.1.2 Confirmed send service

```
ConfirmedSend-RequestPDU ::= SEQUENCE {
      [APPLICATION 0] AddressAREP,
      InvokeID,
      ConfirmedServiceRequest
}

ConfirmedSend-ResponsePDU ::= SEQUENCE {
      [APPLICATION 1] AddressAREP,
      InvokeID,
      pduBody     CHOICE {
         ConfirmedServiceResponse,
         ConfirmedServiceError
      }
}
```

4.1.3 Unconfirmed send service

```
UnconfirmedSend-PDU ::= SEQUENCE {
      [APPLICATION 2] AddressAREP,
      InvokeID,
      pduBody     CHOICE {
         UnconfirmedServiceRequest,
         UnconfirmedSendPD-PDU
      }
}
```

4.1.4 Unconfirmed acknowledge send service

```
UnconfirmedAcknowledgeSend-CommandPDU ::= SEQUENCE {
      [APPLICATION 3] AddressAREP,
      InvokeID,
      UnconfirmedServiceRequest
}
```

4.1.5 InvokeID

```
InvokeID ::=  Unsigned8
```

4.1.6 Establish service

```
MaxOSCC ::= Unsigned8

MaxOSCS ::= Unsigned8

MaxUCSC ::= Unsigned8

MaxUCSS ::= Unsigned8

CIU ::= Unsigned32

Establish-RequestPDU ::= SEQUENCE {
      [APPLICATION 4] AddressAREP,
      ConType,
      MaxOSCC,
      MaxOSCS,
      MAXUCSC,
      MAXUCSS,
      CIU,
      InvokeID,
      initiateRequest                    [PRIVATE 0] IMPLICIT Initiate-RequestPDU
}

Establish-ResponsePDU ::= SEQUENCE {
      [APPLICATION 5] AddressAREP,
      InvokeID,
      initiateResponse                   [PRIVATE 0] IMPLICIT Initiate-ResponsePDU
}

Establish-ErrorPDU ::= SEQUENCE {
      [APPLICATION 6] AddressAREP,
      InvokeID,
      initiateError                      [PRIVATE 0] IMPLICIT Initiate-ErrorPDU
}
```

4.1.7 ConType

```
ConType ::= ENUMERATED {
   mmaz          (0)
}
```

4.1.8 Data send acknowledge service

```
DataSendAcknowledge-PDU ::= SEQUENCE {
      Protocol-Code,[APPLICATION 8],Address2ARP,        --- Protocol-Code in the higher nibble of the
octet!!!
      Block-Number,
      Block-Length,
      Protocol-Data
}
```

4.1.9 Protocol-code

```
Protocol-Code ::= ENUMERATED {
   TCP/IP        (1)                  --- TCP/IP protocol
}
```

4.1.10 Block-number

```
Block-Number ::= Unsigned8             --- 0  no blocking
                                           ---- 1...254 block number
                                           ---- 255 last block
```

4.1.11 Block-length

```
Block-Length ::= Unsigned
```

4.1.12 Protocol-data

```
Protocol-Data::= Any                   --- transparent protocol transfer
```

4.1.13 Address2 ARP

```
Address2ARP ::= SEQUENCE {
    Destination-Address,
    Source-Address,
    Destination-Node,
    Destination-Subnode,
    Source-Node,
    Source-Subnode
}
```

4.1.14 Destination-address

```
Destination-Address ::= OctetString                    --- 3 Octets
```

4.1.15 Source-address

```
Source-Address ::= OctetString                         --- 3 Octets
```

4.1.16 Destination-node

```
Destination-Node ::= Unsigned8
```

4.1.17 Source-node

```
Source-Node ::= Unsigned8
```

4.1.18 Destination-subnode

```
Destination-Subnode ::= Unsigned8
```

4.1.19 Source-subnode

```
Source-Subnode ::= Unsigned8
```

4.2 Abstract syntax of PDUBody

4.2.1 Abort service

```
Abort-PDU ::= SEQUENCE {
        [APPLICATION 7] AddressAREP,
        Identifier,
        ReasonCode,
        AdditionalDetail
}
```

4.2.2 ConfirmedServiceRequest

```
ConfirmedServiceRequest::= CHOICE {
    read-Request                    [0] IMPLICIT Read-RequestPDU,
    write-Request                   [3] IMPLICIT Write-RequestPDU,
    start-Request                   [6] IMPLICIT Start-RequestPDU,
    stop-Request                    [9] IMPLICIT Stop-RequestPDU,
    status-Request                  [15] IMPLICIT Status-RequestPDU,
    identify-Request                [18] IMPLICIT Identify-RequestPDU,
    getAttributes1-Request          [21] IMPLICIT GetAttributes-RequestPDU,    -- short form
    getAttributes2-Request          [35] IMPLICIT GetAttributes-RequestPDU,    -- long form
    reset-Request                   [36] IMPLICIT Reset-RequestPDU,
    resume-Request                  [39] IMPLICIT Resume-RequestPDU
}
```

4.2.3 ConfirmedServiceResponse

```
ConfirmedServiceResponse ::= CHOICE {
   read-Response                    [1] IMPLICIT Read-ResponsePDU,
   write-Response                   [4] IMPLICIT Write-ResponsePDU,
   start-Response                   [7] IMPLICIT Start-ResponsePDU,
   stop-Response                   [10] IMPLICIT Stop-ResponsePDU,
   status-Response                 [16] IMPLICIT Status-ResponsePDU,
   identify-Response               [19] IMPLICIT Identify-ResponsePDU,
   getAttributes-Response          [22] IMPLICIT GetAttributes-ResponsePDU,
   reset-Response                  [37] IMPLICIT Reset-ResponsePDU,
   resume-Response                 [40] IMPLICIT Resume-ResponsePDU
}
```

4.2.4 ConfirmedServiceError

```
ConfirmedServiceError ::= CHOICE {
   read-Error                      [2] IMPLICIT ErrorType,
   write-Error                     [5] IMPLICIT ErrorType,
   start-Error                     [8] IMPLICIT ErrorFiType,
   stop-Error                     [11] IMPLICIT ErrorFiType,
   status-Error                   [17] IMPLICIT ErrorType,
   identify-Error                 [20] IMPLICIT ErrorType,
   getAttributes-Error            [23] IMPLICIT ErrorType,
   reset-Error                    [38] IMPLICIT ErrorFiType,
   resume-Error                   [41] IMPLICIT ErrorFiType
}
```

4.2.5 Error type

Error type as specified in 4.1.4.6.

4.2.6 Error Fi type

```
ErrorFiType ::= SEQUENCE {
   errorClass                      [0] IMPLICIT ErrorClass,
   additionalCode                  [1] IMPLICIT Integer16 OPTIONAL,
   fiState                         [3] IMPLICIT Integer8
}
```

4.2.7 Error class

Error Class as specified in 4.1.4.7.

4.2.8 Unconfirmed PDUs

```
UnconfirmedServiceRequest ::= CHOICE {
   informationReport-Request       [12] IMPLICIT InformationReport-RequestPDU,
   reject-Request                  [34] IMPLICIT Reject-RequestPDU
}

UnconfirmedSendPD-PDU ::= BIT STRING
```

4.2.9 Management ASE

4.2.9.1 Get attributes service

```
GetAttributes-Request-PDU ::= SEQUENCE {
   listOfAttributes                [PRIVATE 0] IMPLICIT Mn_SelectedAttributes,
   accessSpecification  CHOICE {
      index                           [1] IMPLICIT Gn_NumericID,
      variableName                 [2] IMPLICIT Gn_Name,
      fiName                       [5] IMPLICIT Gn_Name,
      startIndex                   [7] IMPLICIT Gn_NumericID
   }
}

GetAttributes-ResponsePDU ::= SEQUENCE {
   more                            [PRIVATE 0] IMPLICIT Gn_MoreFollows,
   listOfObjectDefinition          [PRIVATE 1] IMPLICIT SEQUENCE OF Gn_ObjectDefinition
}
```

4.2.10 Application process ASE

4.2.10.1 Get status service

```
Status-RequestPDU ::= NULL

Status-ResponsePDU ::= SEQUENCE {
   logicalStatus                    [PRIVATE 0] IMPLICIT ENUMERATED {
      ready-for-communication          (0),
      limited-services-permitted       (2),
   },
   physicalStatus                   [PRIVATE 1] IMPLICIT ENUMERATED {
      operational                      (0),
      partially-operational            (1),
      inoperable                       (2),
      needs-commissioning              (3)
   },
   localDetail                      [PRIVATE 2] IMPLICIT BitString OPTIONAL
 }
```

4.2.10.2 Identify service

```
Identify-RequestPDU ::= NULL

Identify-ResponsePDU ::= SEQUENCE {
   vendorName                       [PRIVATE 0] IMPLICIT VisibleString,
   modelIdentifier                  [PRIVATE 1] IMPLICIT VisibleString,
   vendorRevision                      [PRIVATE 2] IMPLICIT VisibleString
}
```

4.2.10.3 Initiate service

```
Initiate-RequestPDU ::= SEQUENCE {
   versionObjectDefinitionsCalling  [PRIVATE 0] IMPLICIT Integer16,
   apDescriptorCalling              [PRIVATE 1] IMPLICIT OctetString,
   accessProtectionSupportedCalling [PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
   passwordAndAccessGroupsCalling   [PRIVATE 3] IMPLICIT Ap_AccessControl,
   configuredMaxPduSizeSending      [PRIVATE 4] IMPLICIT Unsigned8,
   configuredMaxPduSizeReceiving    [PRIVATE 5] IMPLICIT Unsigned8,
   listOfSupportedServicesCalling   [PRIVATE 6] IMPLICIT Mn_PduSupportedMap
}

Initiate-ResponsePDU ::= SEQUENCE {
   versionObjectDefinitionsCalled   [PRIVATE 0] IMPLICIT Integer16,
   apDescriptorCalled               [PRIVATE 1] IMPLICIT OctetString,
   accessPrivilegeSupportedCalled   [PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
   passwordAndAccessGroupsCalled    [PRIVATE 3] IMPLICIT Ap_AccessControl
}

Initiate-ErrorPDU ::= SEQUENCE {
   errorCode                        [PRIVATE 0] IMPLICIT  ENUMERATED {
      other                            (0),
      max-fal-pdu-size-insufficient    (1),
      service-not-supported            (2),
      version-obj-def-incompatible     (3),
      user-initiate-denied             (4),
      password-error                   (5),
      profile-number-incompatible      (6)
      },
   maxPduLengthSendingCalled        [PRIVATE 1] IMPLICIT Unsigned8,
   maxPduLengthReceivingCalled      [PRIVATE 2] IMPLICIT Unsigned8,
   listOfSupportedServicesCalled    [PRIVATE 3] IMPLICIT Mn_PduSupportedMap
 }
```

4.2.10.4 Reject service

```
Reject-RequestPDU ::= SEQUENCE {
   original-invokeID                [PRIVATE 0] IMPLICIT Integer8,
   reject-code                      [PRIVATE 1] IMPLICIT ENUMERATED {
      pdu-size                         (5)
   }
}
```

4.2.11 Function invocation ASE

4.2.11.1 Reset service

```
Reset-RequestPDU ::= SEQUENCE {
   keyAttribute                    Gn_KeyAttribute
}

Reset-ResponsePDU ::= NULL
```

4.2.11.2 Resume service

```
Resume-RequestPDU ::= SEQUENCE {
   keyAttribute                    Gn_KeyAttribute
}

Resume-ResponsePDU ::= NULL
```

4.2.11.3 Start service

```
Start-RequestPDU ::= SEQUENCE {
   keyAttribute                    Gn_KeyAttribute
}

Start-ResponsePDU ::= NULL
```

4.2.11.4 Stop service

```
Stop-RequestPDU ::= SEQUENCE {
   keyAttribute                    Gn_KeyAttribute
}

Stop-ResponsePDU ::= NULL
```

4.2.12 Variable ASE

4.2.12.1 Information report service

```
InformationReport-RequestPDU ::=  SEQUENCE {
   index                           Gn_NumericID,
   subIndex                        [PRIVATE 0] IMPLICIT Gn_SubIndex OPTIONAL,
   value                           [PRIVATE 1] IMPLICIT ANY
}
```

4.2.12.2 Read service

```
Read-RequestPDU ::= SEQUENCE {
   index                           Gn_NumericID,
   subIndex                        [PRIVATE 0] IMPLICIT Gn_SubIndex OPTIONAL
}

Read-ResponsePDU ::= SEQUENCE {
   value                           [PRIVATE 0] IMPLICIT ANY
}
```

4.2.12.3 Write service

```
Write-RequestPDU ::= SEQUENCE {
   index                           Gn_NumericID,
   subIndex                        Gn_SubIndex OPTIONAL,
   value                           [PRIVATE 0] IMPLICIT ANY
}

Write-ResponsePDU ::= NULL
```

4.3 Type definitions for ASEs

4.3.1 AP ASE types

4.3.1.1 Ap_AccessProtectionSupported

```
Ap_AccessProtectionSupported ::= Boolean      -- True means Access Protection is supported.
                                              -- False means Access Protection is not supported.
```

4.3.1.2 Ap_AccessControl

```
Ap_AccessControl ::= BitString {          -- The Password (Unsigned8) is encoded as a bit string.
   password_Bit1                          (8),
   password_Bit2                          (7),
   password_Bit3                          (6),
   password_Bit4                          (5),
   password_Bit5                          (4),
   password_Bit6                          (3),
   password_Bit7                          (2),
   password_Bit8                          (1),
   access_Groups-1                        (16),
   access_Groups-2                        (15),
   access_Groups-3                        (14),
   access_Groups-4                        (13),
   access_Groups-5                        (12),
   access_Groups-6                        (11),
   access_Groups-7                        (10),
   access_Groups-8                        (9)
}
```

4.3.2 AR ASE types

4.3.2.1 Reason code and additional detail

4.3.2.1.1 Reason code

```
ReasonCode ::= Unsigned8
```

4.3.2.1.2 Additional detail

```
AdditionalDetaill ::= OctetString
```

4.3.2.2 AREP

```
AddressAREP::= Unsigned8                  -- Least significant octet of DLCEP address
```

4.3.2.3 Abort type identifier

```
Identifier ::= ENUMERATED {
   fal-user                               (0),
   fal-APO-ASE                            (1),
   fal-AR-ASE                             (2),
   dll                                    (3)
}
```

4.3.3 Function Invocation ASE types

4.3.3.1 Fi_AccessPrivilege

```
Fi_AccessPrivilege ::= BitString {
   rightToStartPassword                   (24),
   rightToStopPassword                    (23),
   rightToDeletePassword                  (22),
   rightToStartAccessGroup                (20),
   rightToStopAccessGroup                 (19),
   rightToDeleteAccessGroup               (18),
   rightToStartAllPartner                 (32),
   rightToStopAllPartner                  (31),
   rightToDeleteAllPartner                (30),
   password_Bit1                          (8),     -- The Password (Unsigned8) is encoded as a bit string.
   password_Bit2                          (7),
   password_Bit3                          (6),
   password_Bit4                          (5),
   password_Bit5                          (4),
   password_Bit6                          (3),
   password_Bit7                          (2),
   password_Bit8                          (1),
   access_Groups-1                        (16),
   access_Groups-2                        (15),
   access_Groups-3                        (14),
   access_Groups-4                        (13),
   access_Groups-5                        (12),
   access_Groups-6                        (11),
   access_Groups-7                        (10),
   access_Groups-8                        (9)
```

```
}
```

4.3.3.2 Fi_State

```
Fi_State ::= Unsigned8 {
   unrunnable                    (1),
   idle                          (2),
   running                       (3),
   stopped                       (4),
   starting                      (5),
   stopping                      (6),
   resuming                      (7),
   resetting                     (8)
}
```

4.3.4 Management ASE types

4.3.4.1 Mn_PduSupportedMap

```
Mn_PduSupportedMap ::= BIT STRING {
   getAttributes-RequestPDU      (1),      -- Requester
   start-RequestPDU              (8),
   stop-RequestPDU               (9),
   resume-RequestPDU             (9),
   reset-RequestPDU              (9),
   read-RequestPDU               (11),
   write-RequestPDU              (12),
   informationReport-RequestPDU  (17),
   getAttributes-ResponsePDU     (25),     -- Responder
   start-ResponsePDU             (33),
   stop-ResponsePDU              (33),
   resume-ResponsePDU            (33),
   reset-ResponsePDU             (33),
   read-ResponsePDU              (35),
   write-ResponsePDU             (36),
   informationReport-ResponsePDU (41)
}
```

4.3.5 Variable ASE types

4.3.5.1 Vr_AccessPrivilege

```
Vr_AccessPrivilege ::= BitString {
   rightToReadPassword           (24),
   rightToWritePassword          (23),
   rightToDeletePassword         (22),
   rightToReadAccessGroup        (20),
   rightToWriteAccessGroup       (19),
   rightToDeleteAccessGroup      (18),
   rightToReadAllPartners        (32),
   rightToWriteAllPartners       (31),
   rightToDeleteAllPartners      (30),
   password_Bit1                 (8),      -- The Password (Unsigned8) is encoded as a bit string.
   password_Bit2                 (7),
   password_Bit3                 (6),
   password_Bit4                 (5),
   password_Bit5                 (4),
   password_Bit6                 (3),
   password_Bit7                 (2),
   password_Bit8                 (1),
   access_Groups-1               (16),
   access_Groups-2               (15),
   access_Groups-3               (14),
   access_Groups-4               (13),
   access_Groups-5               (12),
   access_Groups-6               (11),
   access_Groups-7               (10),
   access_Groups-8               (9)
}
```

4.3.6 General types

4.3.6.1 Gn_Deletable

```
Gn_Deletable ::= Boolean          -- True means deletable.
                                  -- False means not deletable.
```

4.3.6.2 Gn_KeyAttribute

```
Gn_KeyAttribute ::= CHOICE {
-- When this type is specified, only the key attributes of the class referenced are valid.
   numericID                    [0] IMPLICIT Gn_NumericID,
   name                         [1] IMPLICIT Gn_Name,
   listName                     [2] IMPLICIT Gn_Name,
   numericAddress               [4] IMPLICIT Gn_NumericAddress,
   symbolicAddress              [5] IMPLICIT Gn_SymbolicAddress
}
```

4.3.6.3 Gn_Length

```
Gn_Length ::= Unsigned8
```

4.3.6.4 Gn_MoreFollows

```
Gn_MoreFollows ::= Boolean
```

4.3.6.5 Gn_NumericID

```
Gn_NumericID ::= Unsigned16          -- The values of this parameter are unique within an AP.
```

4.3.6.6 Gn_Name

```
Gn_Name ::= OctetString
```

4.3.6.7 Gn_ObjectClass

```
Gn_ObjectClass ::= ENUMERATED {
   functionInvocation           (3),
   fixedLengthStringDataType    (5),
   structuredDataType           (6),
   fixedLengthStringVariable    (7),
   arrayVariable                (8),
   dataStructureVariable        (9),
}
```

4.3.6.8 Gn_ObjectDefinition

```
Gn_ObjectDefinition ::= OctetString  -- The semantics of this parameter are application specific.
```

4.3.6.9 Gn_Reusable

```
Gn_Reusable ::= Boolean              -- True means reusable.
                                     -- False means not reusable.
```

4.3.6.10 Gn_SubIndex

```
Gn_SubIndex ::= Unsigned8
```

4.3.6.11 Gn_TypeDescription

```
Gn_TypeDescription ::= CHOICE {
   boolean                      [1] Gn_Length,
   integer8                     [2] Gn_Length,
   integer16                    [3] Gn_Length,
   integer32                    [4] Gn_Length,
   unsigned8                    [5] Gn_Length,
   unsigned16                   [6] Gn_Length,
   unsigned32                   [7] Gn_Length,
   float                        [8] Gn_Length,
   visiblestring                [9] Gn_Length,
   octetstring                  [10] Gn_Length,
   binaryDate                   [11] Gn_Length,
   timeOfDay                    [12] Gn_Length,
   timeDifference               [13] Gn_Length,
   bitstring                    [14] Gn_Length
}
```

4.3.7 Object definitions

4.3.7.1 Top Level Definition

```
Object-Definition ::= CHOICE {
        [PRIVATE 0] ListHeader,
        [PRIVATE 1] DataTypeList,
        [PRIVATE 2] StaticList,
        [PRIVATE 3] FunctionInvocationDefinition
}
```

4.3.7.2 ListHeader

```
ListHeader ::= SEQUENCE {
   numericId                                    [PRIVATE 0] IMPLICIT Unsigned16,
   romRamFlag                                   [PRIVATE 1] IMPLICIT Boolean,
   maxNameLength                                [PRIVATE 2] IMPLICIT Unsigned8,
   accessProtectionSupported                    [PRIVATE 3] IMPLICIT Boolean,
   versionOfObjectDefinition                    [PRIVATE 4] IMPLICIT Integer16,
   localReferenceOfListHeader                   [PRIVATE 5] IMPLICIT Unsigned32 OPTIONAL,
   numberOfEntriesInDataTypeList                [PRIVATE 6] IMPLICIT Unsigned16,
   localReferenceOfDataTypeList                 [PRIVATE 7] IMPLICIT Unsigned32 OPTIONAL,
   firstNumericIdOfStaticList                   [PRIVATE 8] IMPLICIT Unsigned16,
   numberOfEntriesInStaticList                  [PRIVATE 9] IMPLICIT Unsigned16,
   localReferenceOfStaticList                   [PRIVATE 10] IMPLICIT Unsigned32 OPTIONAL,
   firstNumericIdOfVariableListDefinition       [PRIVATE 11] IMPLICIT Unsigned16,
   numberOfEntriesInVariableListDefinition          [PRIVATE 12] IMPLICIT Unsigned16,
   localReferenceOfVariableListDefinition       [PRIVATE 13] IMPLICIT Unsigned32 OPTIONAL,
   firstNumericIdOfFunctionInvocationDefinition [PRIVATE 14] IMPLICIT Unsigned16,
   numberOfEntriesInFunctionInvocationDefinition [PRIVATE 15] IMPLICIT Unsigned16,
   localReferenceOfFunctionInvocationDefinition [PRIVATE 16] IMPLICIT Unsigned32 OPTIONAL
}
```

4.3.7.3 DataTypeList

```
DataTypeList ::= CHOICE {
        [PRIVATE 0] DataTypeDefinition,
                [PRIVATE 1] StructuredDataTypeDefinition
}

DataTypeDefinition ::= SEQUENCE {
   numericId                                    Gn_NumericID,
   objectClass                                  Gn_ObjectClass,
   dataTypeNameLength                           Gn_Length,
   dataTypeName                                 [PRIVATE 0] IMPLICIT VisibleString OPTIONAL
}

StructuredDataTypeDefinition ::= SEQUENCE {
   numericId                                    Gn_NumericID,
   objectClass                                  Gn_ObjectClass,
   numberOfElements                         [PRIVATE 0] IMPLICIT Integer8,
   recordList      SEQUENCE OF SEQUENCE{
      numericIdOfDataTypeDefinition             Gn_NumericID,
      dataLength                                Gn_Length
   }
}
```

4.3.7.4 StaticList

```
StaticList ::= CHOICE {
        [PRIVATE 0] VariableDefinition,
        [PRIVATE 1] ArrayDefinition,
        [PRIVATE 2] StructureDefinition
}

VariableDefinition ::= SEQUENCE {
   numericId                                    Gn_NumericID,
   objectClass                                  Gn_ObjectClass,
   numericIdOfDataTypeDefinition                Gn_NumericID,
   dataLength                                   Gn_Length,
   accessPrivilege                              Vr_AccessPrivilege OPTIONAL,
   localReferenceOfVariable                 [PRIVATE 0] IMPLICIT Unsigned32 OPTIONAL,
   variableName                             [PRIVATE 1] IMPLICIT VisibleString OPTIONAL,
   extension                                [PRIVATE 2] IMPLICIT OctetString OPTIONAL
}
```

```
ArrayDefinition ::= SEQUENCE {
   numericId                          Gn_NumericID,
   objectClass                        Gn_ObjectClass,
   numericIdOfDataTypeDefinition      Gn_NumericID,
   dataLength                         Gn_Length,
   numberOfElements                   [PRIVATE 0] IMPLICIT Unsigned8,
   accessPrivilege                    Vr_AccessPrivilege OPTIONAL,
   localReferenceOfArray              [PRIVATE 1] IMPLICIT Unsigned32 OPTIONAL,
   arrayName                          [PRIVATE 2] IMPLICIT VisibleString OPTIONAL,
   extension                          [PRIVATE 3] IMPLICIT OctetString OPTIONAL
}

StructureDefinition ::= SEQUENCE {
   numericId                          Gn_NumericID,
   objectClass                        Gn_ObjectClass,
   numericIdOfDataTypeDefinition      Gn_NumericID,
   accessPrivilege                    Vr_AccessPrivilege OPTIONAL,
   structureName                      [PRIVATE 0] IMPLICIT VisibleString OPTIONAL,
   extension                          [PRIVATE 1] IMPLICIT OctetString OPTIONAL,
   localReferenceOfElement            [PRIVATE 2] IMPLICIT SEQUENCE OF Unsigned32 OPTIONAL
}
```

4.3.7.5 FunctionInvocationDefinition

```
FunctionInvocationDefinition ::= SEQUENCE {
   numericId                          Gn_NumericID,
   objectClass                        Gn_ObjectClass,
   numberOfRelatedObjects             [PRIVATE 0] IMPLICIT Unsigned8,
   accessPrivilege                    Fi_AccessPrivilege OPTIONAL,
   deletable                          Gn_Deletable,
   reusable                           Gn_Reusable,
   functionInvocationState            FI_State,
   numericIdOfLoadRegion              SEQUENCE OF Gn_NumericID,
   functionInvocationName             [PRIVATE 1] IMPLICIT VisibleString OPTIONAL,
   extension                          [PRIVATE 2] IMPLICIT OctetString OPTIONAL
}
```

4.4 Abstract syntax of data types

4.4.1 Referenced data types

4.4.2 Notation for the Boolean type

```
Boolean ::= BOOLEAN                   -- TRUE if the value is non-zero.
                                      -- FALSE if the value is zero.
```

4.4.3 Notation for the Integer types

```
Integer ::= INTEGER                   -- any integer
Integer8 ::= INTEGER  (-128..+127)    -- range  -2^7 <= i <= 2^7-1
Integer16 ::= INTEGER  (-32768..+32767) -- range  -2^15 <= i <= 215-1
Integer32 ::= INTEGER                 -- range  -2^31 <= i <= 2^31-1
```

4.4.4 Notation for the Unsigned types

```
Unsigned ::= INTEGER                  -- any non-negative integer
Unsigned8 ::= INTEGER  (0..255)       -- range 0 <= i <= 2^8-1
Unsigned16 ::= INTEGER  (0..65535)    -- range 0 <= i <= 2^16-1
Unsigned32 ::= INTEGER                -- range 0 <= i <= 2^32-1
```

4.4.5 Notation for the Floating Point type

```
Floating32 ::= BIT STRING  SIZE (4)   --  ISO/IEC/IEEE 60559 Single precision
```

4.4.6 Notation for the BitString type

```
BitString ::= BIT STRING              -- For generic use
BitString8 ::= BIT STRING  SIZE (8)   -- Fixed eight bits bitstring
BitString16 ::= BIT STRING  SIZE (16) -- Fixed 16 bits bitstring
BitString32 ::= BIT STRING  SIZE (32) -- Fixed 32 two bits bitstring
```

4.4.7 Notation for the OctetString type

```
OctetString ::= OCTET STRING                -- For generic use
OctetString2 ::= OCTET STRING  SIZE (2)     -- Fixed two-octet octet string
OctetString4 ::= OCTET STRING  SIZE (4)     -- Fixed four-octet octet string
OctetString6 ::= OCTET STRING  SIZE (6)     -- Fixed six-octet octet string
OctetString7 ::= OCTET STRING  SIZE (7)     -- Fixed seven-octet octet string
OctetString8 ::= OCTET STRING  SIZE (8)     -- Fixed eight-octet octet string
OctetString16 ::= OCTET STRING  SIZE (16)   -- Fixed 16 octet octet string
```

4.4.8 Notation for VisibleString type

```
VisibleString2 ::= VisibleString  SIZE (2)      -- Fixed two-octet visible string
VisibleString4 ::=VisibleString  SIZE (4)       -- Fixed four-octet visible string
VisibleString8 ::= VisibleString  SIZE (8)      -- Fixed eight-octet visible string
VisibleString16 ::= VisibleString  SIZE (16)    -- Fixed 16 octet visible string
```

4.4.9 Notation for BinaryDate type

BinaryDate ::= OctetString7

4.4.10 Notation for TimeOfDay type

TimeOfDay ::= OctetString6

4.4.11 Notation for TimeDifference type

TimeDifference ::= OctetString6

4.4.12 Notation for TimeValue type

TimeValue ::= OctetString8

4.4.13 Nortation for DL—Time-offset type

DL-Time-offsetType ::= OctetString

5 Transfer syntax

5.1.1 General

Additional information is to be added to the user data to allow an unique association of the data at the communication partner. The coding rules for the additional information are optimized to produce messages as short as possible in accordance with fieldbus requirements. The frequency of occurrence of special messages is taken into account.

The conditions in the fieldbus area are the following:

- short messages
- low number of different messages
- some messages such as Read and Write occur especially often.

5.1.2 Coding rules

The structuring of a FMS PDU is done either by inserting explicitly Identification Information or by implicit agreements.

Structure of a PDU is shown in Figure 1.

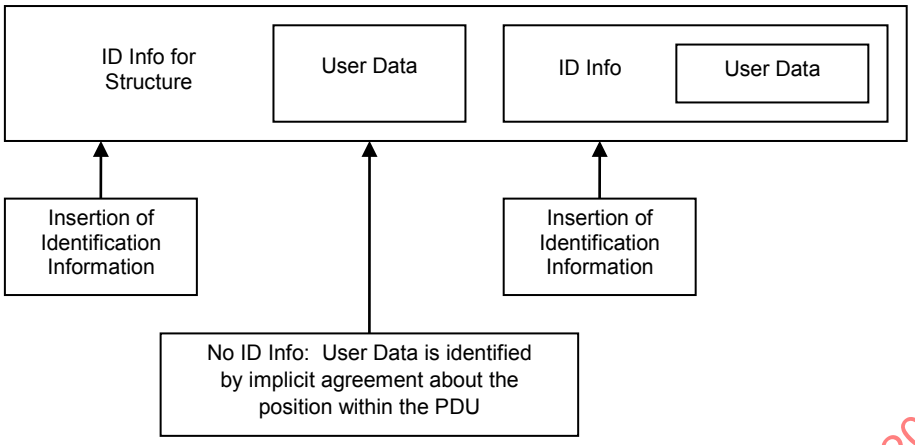


Figure 1 – Insertion of identification information in the FMS PDU

The Identification Information consists of P/C flag, tag and length. The structures and the user data of the PDU may be identified using this Identification Information.

The semantics of the user data are either known from the context or are universally known (context specific tags or universal tags). In the syntax description the context specific tags are enclosed in rectangular brackets. If the semantics of a parameter are implicitly known from the position in the PDU, no tag is used.

The following restrictions on the usage of implicitly known components (universal tags) shall be made:

- the length of the user data shall be fixed,
- the component may not be OPTIONAL,
- the component may not be inside a CHOICE construct.

5.1.3 Structure of the identification information

5.1.3.1 General

The Identification Information (ID Info) consists of P/C flag, tag and length, as shown in Figure 2.

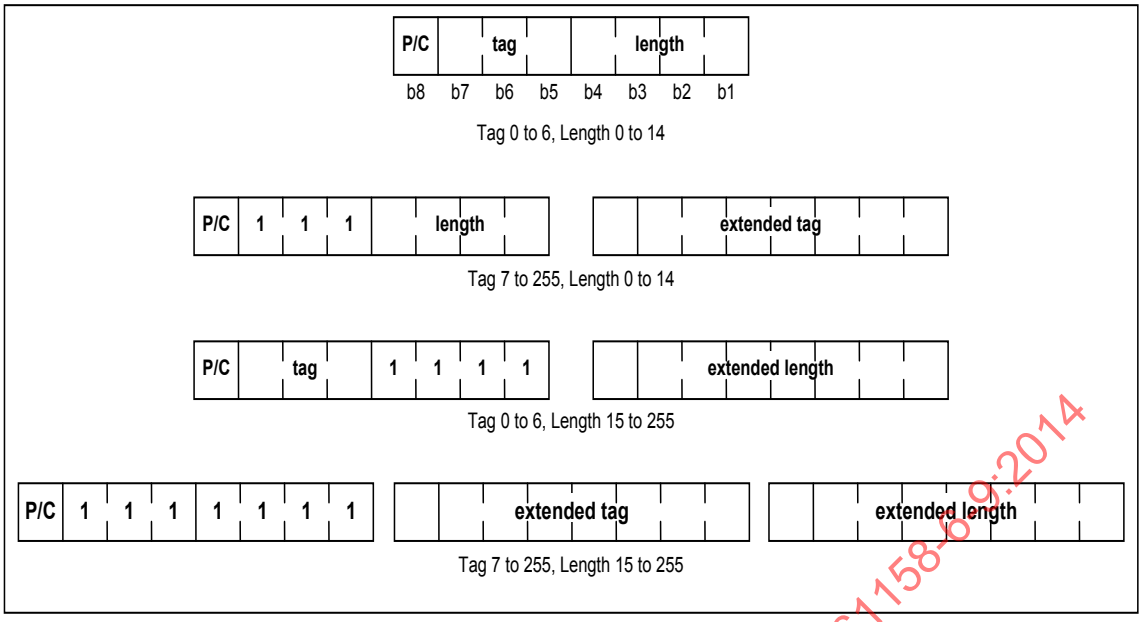


Figure 2 – Identification

The P/C flag indicates if it is a simple component (primitive) or if it is a structured component (constructed, SEQUENCE, SEQUENCE OF).

P/C Flag =0 <=> simple component
P/C Flag =1 <=> structured component

The tag identifies the semantics of the component.

The length is the length of the component in octets if this component is a simple one or the number of contained components if this component is structured.

The whole ID Info is coded in one octet, if possible. The octet is divided into 3 parts of different lengths.

- P/C flag 1 bit
- tag 3 bits
- length 4 bits

If the space in the fields for the length or for the tag is not sufficient, an extension is used. For this all bits of the concerned field are set to one. The information is then encoded in the following octet. The range of the tag is 0 to 6 and the range of the length is 0 to 14 when no extension is used. These ranges are preferred for the most frequently occurring messages because they produce very short PDUs.

If an extension has to be used for both the tag and the length, the tag is encoded in the first subsequent octet and the length is encoded in the second subsequent octet.

5.1.3.2 Data types

5.1.3.2.1 General

User data is always a simple (primitive) component. It is encoded as shown in Figure 3.

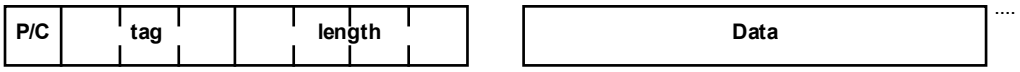


Figure 3 – Coding with identification

If the semantics of the user data are known implicitly from the position in the PDU and the length is fixed and implicitly known, then no Identification Information is added as shown in Figure 4.

Figure 4 – Coding without identification

5.1.3.2.2 Coding for Boolean type

Representation of the value true or false in one octet as shown in Figure 5 and Figure 6.

Notation: Range of values:	Boolean
Coding:	true or false false is represented by the value 0, true is represented by the value FF

bits	8	7	6	5	4	3	2	1
octet 1	1	1	1	1	1	1	1	1

Figure 5 – Representation of the value true

bits	8	7	6	5	4	3	2	1
octet 1	0	0	0	0	0	0	0	0

Figure 6 – Representation of the value false

5.1.3.2.3 Coding for Integer types

Integer values are signed quantities as shown in Figure 7.

Notation:	**Integer8, Integer16, Integer32**		
Range of Values:	Data type	range of values	length
	Integer8	$-128 \le i \le 127$	1 octet
	Integer16	$-32768 \le i \le 32767$	2 octets
	Integer32	$-2^{31} \le i \le 2^{31} - 1$	4 octets
Coding:	In two's complement representation the MSB (Most Significant Bit) is the bit after the sign bit (SN) in the first octet. SN = 0: positive numbers and z SN = 1: negative numbers		

bits	8	7	6	5	4	3	2	1
octets								
1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Figure 7 – Coding of data of data type Integer16

5.1.3.2.4 Coding for Unsigned types

Unsigned Values are encoded as shown in Figure 8.

Notation:	**Unsigned8, Unsigned16, Unsigned32**		
Range of Values:	**Data type**	**range of values**	**length**
	Unsigned8	$0 \le i \le 255$	1 octet
	Unsigned16	$0 \le i \le 65\,535$	2 octets
	Unsigned32	$0 \le i \le$ 4 294 967 295	4 octets
Coding	**Binary**		

bits	8	7	6	5	4	3	2	1
octets								
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Figure 8 – Coding of data of data type Unsigned16

5.1.3.2.5 Coding for Floating Point type

Floating Point values are encoded as shown in Figure 9.

Notation:	**Floating-Point (4 octet)**
Range of Values:	see IEEE Std 754 Short Real Number (32 bits)
Coding:	see IEEE Std 754 Short Real Number (32 bits)

								LSB
bits	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
octets	**Exponent (E)**							
1	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	(E)	**Fraction (F)**						
2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
	Fraction (F)							
3	2^{-8}	2^{-9}	2-10	2-11	2-12	2-13	2-14	2-15
	Fraction (F)							
4	2-16	2-17	2-18	2-19	2-20	2-21	2-22	2^{-23}

SN: sign 0 = positive, 1 = negative

Figure 9 – Coding of data of data type Floating Point

5.1.3.2.6 Coding for Visible String type

Visible String values are encoded as shown in Figure 10.

Notation:	**Visible-String**
Range of Values:	see ISO/IEC 646 and ISO/IEC 2375: Defining
registration	
	number 2 + SPACE
Coding:	see ISO/IEC 646

bits	8	7	6	5	4	3	2	1
octets								
1	first character							
2	second character							
.	etc.							
.								
n								

Figure 10 – Coding of data of data type Visible String

5.1.3.2.7 Coding for Octet String type

Octet String values are encoded as shown in Figure 11.

Notation: Octet String
Coding: Binary

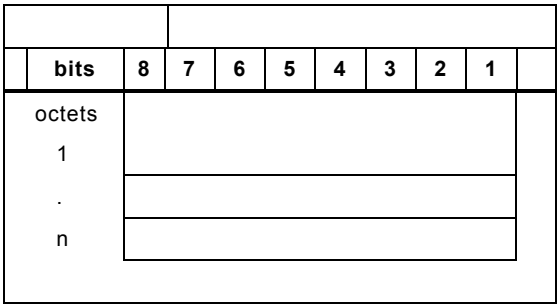


Figure 11 – Coding of data of data type Octet String

5.1.3.2.8 Coding for Date type

The data type Date consists of a calendar date and a time as shown in Table 2 and Figure 12.

Notation: Date/Time
Range of Values: ms to 99 years
Coding: in 7 octets

Table 2 – Coding for Date type

Parameter	Range of values	Meaning of the parameters
ms	0…59 999	milliseconds
min	0…59	minutes
SU	0,1	0: standard time, 1: summer time
RSV	—	reserve
h	0…23	hours
d. of w.	1…7	day of week: 1 = Monday, 7 = Sunday
d. of m.	1…31	day of month
months	1…12	months
years	0…99	years (without the century)

bits	8	7	6	5	4	3	2	1	
octets									
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 999 ms
3	RSV	RSV	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 min
4	SU	RSV	RSV	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…23 h
	day of week			day of month					1…7 d. of w.
5	2^{2}	2^{1}	2^{0}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…31 d. of m.
6	RSV	RSV	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…12 months
7	RSV	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0 … 99 years
	msb								

Figure 12 – Coding of data of type Date

5.1.3.2.9 Coding for Time-of-day type

The data type Time-of-day consists of a time and an optional date.

The time is stated in milliseconds since midnight. At midnight the counting starts with the value zero.

The date is stated in days relatively to the first of January 1984. On the first of January 1984 the date starts with the value zero.

Notation: Time-of-day

Range of Values: $0 \le i \le (2^{28} -1)$ ms
$0 \le i \le (2^{16} -1)$ days

Coding:

The time is represented as a 32 bit binary value. The first four (MSB) bits shall have the value zero.

The (optional) date is encoded as a 16 bit (2 octets) binary value.

The Time-of-day is a string of 4 or 6 octets as shown in Figure 13.

bits	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	number of
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	milliseconds since
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	midnight
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	number of days since 1984-01-01
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	optional
	msb								

Figure 13 – Coding of data of data type Time-of-day

5.1.3.2.10 Coding for Time-difference type

The data type Time-difference consists of a time in milliseconds and an optional day count. The structure is equivalent to the structure of the Time-of-day but it states in this case a Time Difference.

Notation Time-difference

Range of Values: $0 \le i \le (2^{28} - 1)$ ms
$0 \le i \le (2^{16} - 1)$ days

Coding:

The time is represented as a 32 bit binary value. The first four (MSB) bits shall have the value zero. The (optional) date is encoded as a 16 bit (2 octets) binary value. The Time Difference is a string of 4 or 6 octets as shown in Figure 14.

bits	8	7	6	5	4	3	2	1	
octets									
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	number of
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	milliseconds
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	number of days
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	optional
	msb								

Figure 14 – Coding of data of data type Time-difference

5.1.3.2.11 Coding for Bit String type

Figure 15 shows the numbering scheme of the bits of the data type Bit String.

Only multiples of eight are legal values of the length (in bits) of the Bit String.

Notation: Bit String
Coding: Binary

bits	8	7	6	5	4	3	2	1	
octets									
1	0	1	2	3	4	5	6	7	
2	8	9	10	11	12	13	14	15	
.	etc.								
n									

Figure 15 – Coding of data of data type Bit String

5.1.3.2.12 Coding for Time-value type

This data type is used to represent date and time in the required precision for application clock synchronization. It is a 64-bit unsigned fixed-point number, which is the time in 1/32s of a millisecond. When SN bit is 1, the data is negative and in two's complement representation, while the data is positive when SN bit is 0. See Figure 16.

bits	8	7	6	5	4	3	2	1	
octets	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}	
1									
2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}	
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}	
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}	
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	signed integer
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	of 8 octets length of 1/32ms unit
	msb								

Figure 16 – Coding of data of data type Time-value

5.1.3.3 User data definitions

5.1.3.3.1 General

User data is always a simple (primitive) component. It is encoded as shown in Figure 16. The P/C, tag, and length are encoded as shown in Figure 17.

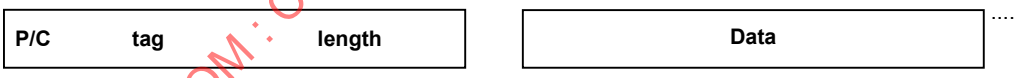


Figure 17 – Coding of data of user data definitions with identifier

If the semantics of the user data are known implicitly from the position in the PDU and the length is fixed and implicitly known, then no Identification Information is added as shown in Figure 18.

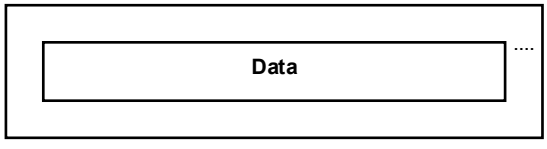


Figure 18 – Coding of data of user data definitions without identifier

Two special definitions are used to describe user data in the PDUBody definitions, Null and Packed. They are defined below.

5.1.3.3.2 Coding for Null

Null has the length zero. There are no subsequent (empty) octets.

5.1.3.3.3 Coding for Packed

Packed contains one or more data elements of the Data types which are chained together without any gap. The composition is known by the user.

5.1.3.4 Coding of structure information

5.1.3.4.1 General

User data may be combined to structured (constructed) components.

The communication partner shall be able to identify these structures and the components of these structures. The P/C flag of the ID Info is 1.

5.1.3.4.2 SEQUENCE

The SEQUENCE structure is comparable with a record. It represents a collection of user data of the same or of different Data types. Before the SEQUENCE structure there is an ID Info, which conveys the length not in octets but in number of components. The number of components is less than the total length in octets. In most cases an extension is not necessary due to this length encoding. Components should be encoded in the order of ANS.1. See Figure 19. Neither duplication nor deletion of an item is allowed.

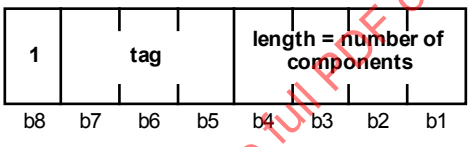


Figure 19 – Coding of ID info for a SEQUENCE

A structure may contain user data or further structures as components. Single components may be OPTIONAL, i.e. they may be omitted. In this case the ID Info is omitted too. A SEQUENCE shall be counted as a single component even if it contains several components.

Example:

The hexadecimal notation is used for the following example of encoding. The upper case letter X is used as a fill-in for unknown values, such as the length of the single components or the tag of the structure. A lower case letter x represents user data.

Syntax Description	Code	comment
Person [1] IMPLICIT SEQUENCE	{94	4 components
[0] Surname,	0X xx ...	tag 0
[1] First name,	1X xx ...	tag 1
[2] City,	2X xx ...	tag 2
[3] Street	3X xx ...	tag 3
}		

5.1.3.4.3 SEQUENCE OF

The SEQUENCE OF structure represents a succession of components. It is comparable with an array.

The structure may contain one or more components. The components may be user data or structures.

The coding is as for the structure SEQUENCE. For the statement of the number of components the number of repetitions shall be taken into account. The tags of the Syntax Description shall be used for the components of SEQUENCE OF. The same tags may be coded several times in succession.

Example:

```
employeedata [2] IMPLICIT SEQUENCE OF {
   [0] Person
}
```

5.1.3.4.4 CHOICE

A CHOICE represents a selection from a set of predefined possibilities. The components of a CHOICE construct shall have different tags to allow proper identification. Instead of the CHOICE construct the actually selected component is encoded. Only one component should be encoded for a CHOICE. Its tag has a value specified in ASN.1.

Example:

```
Data ::= CHOICE {
   [0] Employeedata,
   [1] Clientdata,
   [2] Supplierdata
}
```

6 Structure of FAL protocol state machines

Interface to FAL services and protocol machines are specified in this clause.

NOTE The state machines specified in Clause 6 and ARPMs defined in Clause 9 only define the valid events for each. It is a local matter to handle these invalid events.

The behavior of the FAL is described by three integrated protocol machines. Specific sets of these protocol machines are defined for different AREP types. The three protocol machines are: 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 20.

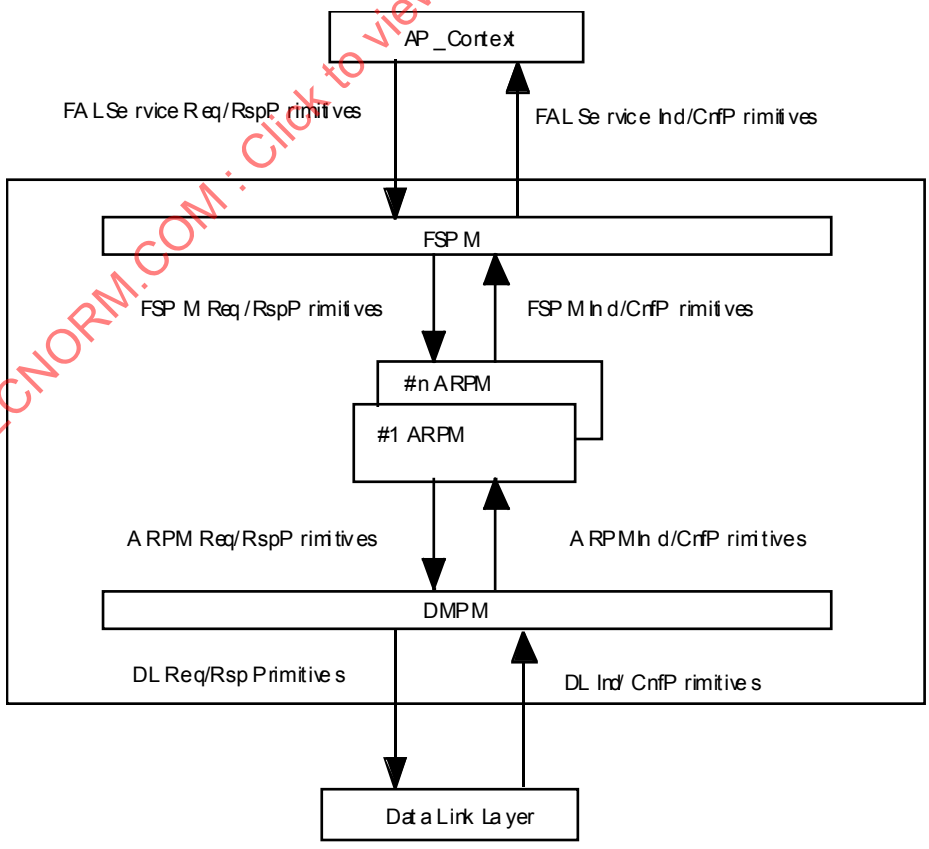


Figure 20 – Relationships among protocol machines and adjacent layers

The FSPM describes the service interface between the AP-Context and a particular AREP. The FSPM is common to all the AREP classes and does not have any state changes. The FSPM is responsible for the following activities:

a) to accept service primitives from the FAL service user and convert them into FAL internal primitives;

b) to select an appropriate ARPM state machine based on the AREP Identifier parameter supplied by the AP-Context and send FAL internal primitives to the selected ARPM;

c) to accept FAL internal primitives from the ARPM and convert them into service primitives for the AP-Context;

d) to deliver the FAL service primitives to the AP-Context based on the AREP Identifier parameter associated with the primitives.

The ARPM describes the establishment and release of an AR and exchange of FAL-PDUs with a remote ARPM(s). The ARPM is responsible for the following activities:

e) to accept FAL internal primitives from the FSPM and create and send other FAL internal primitives to either the FSPM or the DMPM, based on the AREP and primitive types;

f) to accept FAL internal primitives from the DMPM and send them to the FSPM as a form of FAL internal primitives;

g) if the primitives are for the Establish or Abort service, it shall try to establish or release the specified AR.

The DMPM describes the mapping between the FAL and the DLL. It is common to all the AREP types and does not have any state changes. The DMPM is responsible for the following activities:

h) to accept FAL internal primitives from the ARPM, prepare DLL service primitives, and send them to the DLL;

i) to receive DLL indication or confirmation primitives from the DLL and send them to the ARPM in a form of FAL internal primitives.

7 AP-Context state machines

7.1 VCR PM structure

This fieldubs defines a single state machine, the VCR Connection protocol machine. Its relationship with its user and with the underlying FSPM, as well as primitives exchanged among them, are depicted in Figure 21.

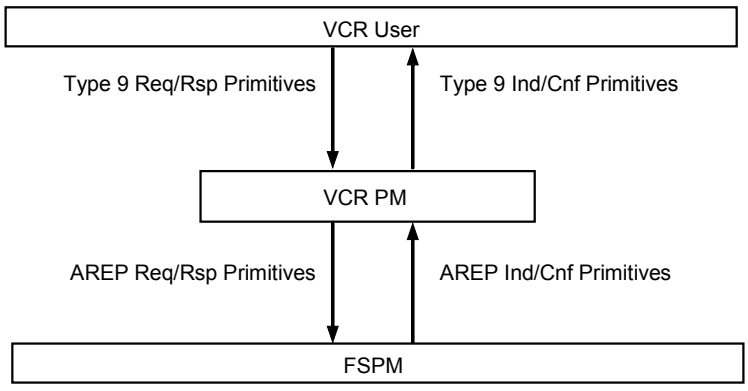


Figure 21 – Relationships among protocol machines and adjacent layers

The VCR Protocol Machine (VCR PM) is an VCR PM state machine that describes the operation of VCRs that are dynamically established by exchanging Initiate service APDUs. The VCR PM is responsible for the following activities:

1. to accept service request and response primitives from the VCR service user;
2. to accept service indication and confirmation primitives from the ARPMs;
3. to deliver service indication and confirmation primitives to the VCR user.
4. to issue service request and response primitives to the ARPM based on the rules of its state machine.

7.2 VCR PM state machine

7.2.1 General

The VCR PM state machine, illustrated in Figure 22, describes the VCR endpoint behavior. For connectionless VCRs, the state machine begins in the OPEN state.

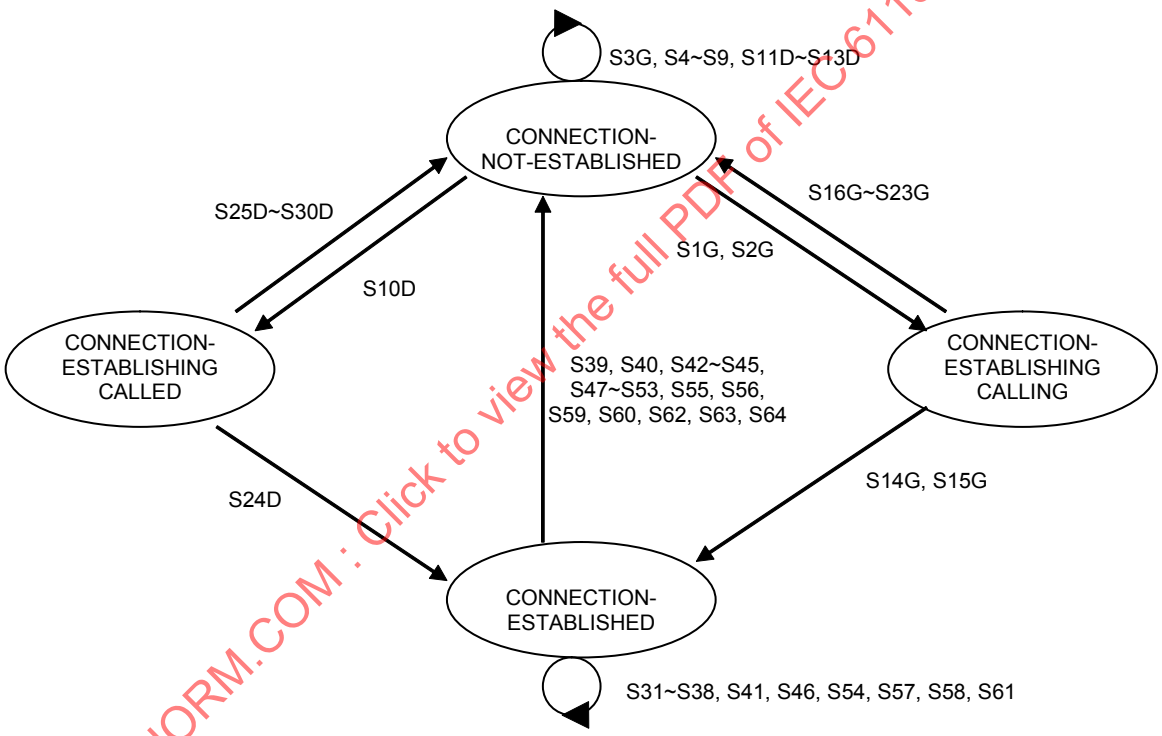


Figure 22 – VCR state machine

NOTE The transaction numbers with “G” suffix are for the calling protocol machine and those with “D” suffix are for the called protocol machine. The transactions without a suffix apply to both protocol machines, or those whose role is yet to be determined.

7.2.2 AP-VCR states

CONNECTION-NOT-ESTABLISHED

The connection is not established. Only the service primitives Initiate.req, ASC.ind, Abort.req and ABT.ind are allowed. All other services shall be rejected with the Abort service.

CONNECTION-ESTABLISHING (CALLING)

The local FMS user wishes to establish the connection. Only the service primitives ASC.cnf(+), ASC.cnf(-), Abort.req and ABT.ind are allowed. All other services shall be rejected with the Abort service. This state is abbreviated as CON-ESTABLISHING-CALLING.

CONNECTION-ESTABLISHING (CALLED)

The remote FMS user wishes to establish the connection. Only the service primitives Initiate.rsp(+), Initiate.rsp(-), Abort.req and ABT.ind are allowed. All other services shall be rejected with the Abort service. This state is abbreviated as CON-ESTABLISHING-CALLED.

CONNECTION-ESTABLISHED

The VCR is established. The service primitives Initiate.req, Initiate.rsp(+), Initiate.rsp(-) are not allowed and shall be rejected with the Abort service. The following actions shall be taken to reset a VCR (Reset VCR).

- Set attribute "Outstanding services Counter Sending" and attribute "Outstanding services Counter Receiving" of the VCR (dynamic part) to 0.
- Set state of the VCR to "CONNECTION-NOT-ESTABLISHED".

7.2.3 AP-VCR initiation state transitions

Table 3 defines the AP-VCR state transitions.

Table 3 – AP-VCR state machine transactions

#	Current state	Event or condition => action	Next state

S1G	CONNECTION-NOT-ESTABLISHED	Initiate.req
		&& VCR is valid
		&& VCR type = QUB
		=>
		ASC.req{
		Data := Initiate-RequestPDU
		}
			CONNECTION – ESTABLISHING-CALLING

S2G	CONNECTION-NOT-ESTABLISHED	Initiate.req
		&& VCR is valid
		&& VCR type <> QUB
		=>
		ASC.req{
		Data := NULL
		}
			CONNECTION – ESTABLISHING-CALLING

S3G	CONNECTION-NOT-ESTABLISHED	Initiate.req
		&& VCR is not valid
		=>
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := VCR error
		}
			CONNECTION-NOT-ESTABLISHED

S4	CONNECTION-NOT-ESTABLISHED	Abort.req => (no actions taken)	CONNECTION-NOT-ESTABLISHED

S5	CONNECTION-NOT-ESTABLISHED	FMS service.primitive other than Initiate & Abort
		=>
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := User error
		}
			CONNECTION-NOT-ESTABLISHED

S6	CONNECTION-NOT-ESTABLISHED	ABT.ind => (no actions taken)	CONNECTION-NOT-ESTABLISHED

S7	CONNECTION-NOT-ESTABLISHED	Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FAS error
		}
			CONNECTION-NOT-ESTABLISHED

S8	CONNECTION-NOT-ESTABLISHED	FAS service primitive other than ASC.ind and ABT.ind
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := Connection state conflict FAS
		}
			CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state

S9	CONNECTION-NOT-ESTABLISHED	ASC.ind
		&& Data = not allowed, unknown or faulty FMS PDU
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FMS PDU error
		}
			CONNECTION-NOT-ESTABLISHED

S10D	CONNECTION-NOT-ESTABLISHED	ASC.ind
		&& Data Initiate-RequestPDU
		&& VCR is valid
		&& Max FMS PDU length test is positive
		&& FmsFeaturesSupported test is positive
		=>
		Initiate.ind{
		}
			CONNECTION – ESTABLISHING-CALLED

S11D	CONNECTION-NOT-ESTABLISHED	ASC.ind
		&& Data = Initiate-RequestPDU
		&& VCR is invalid
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := VCR error
		}
			CONNECTION-NOT-ESTABLISHED

S12D	CONNECTION-NOT-ESTABLISHED	ASC.ind
		&& Data = Initiate-RequestPDU
		&& VCR is valid
		&& Max FMS PDU length test is negative
		=>
		ASC.rsp(-){
		Data := Initiate-ErrorPDU{
		ErrorCode := max-fms-pdu-size-insufficient
		}
		}
			CONNECTION-NOT-ESTABLISHED

S13D	CONNECTION-NOT-ESTABLISHED	ASC.ind
		&& Data = Initiate-RequestPDU
		&& VCR is valid
		&& Max FMS PDU length test is positive
		&& FmsFeaturesSupported test is negative
		=>
		ASC.rsp(-){
		Data := Initiate-ErrorPDU{
		ErrorCode := feature-not-supported
		}
		}
			CONNECTION-NOT-ESTABLISHED

S14G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(+)
		&& Data = Initiate-ResponsePDU
		&& VCR type = QUB
		=>
		Initiate.cnf(+){
		}
			CONNECTION-ESTABLISHED

S15G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(+)
		&& Data = NULL
		&& VCR type = QUU or BNU
		=>
		Initiate.cnf(+){
		}
			CONNECTION-ESTABLISHED

S16G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(-) && Data = Initiate-ErrorPDU && VCR type = QUB => Initiate.cnf(-){ ErrorCode := Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED
S17G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(-) && Data = NULL && VCR type = QUU or BNU => Initiate.cnf(-){ ErrorCode := Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	**Event** or **condition** **=> action**	Next state
S18G	CONNECTION - ESTABLISHING -CALLING	ABT.ind => Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION- NOT- ESTABLISHED
S19G	CONNECTION - ESTABLISHING -CALLING	Abort.req => ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION- NOT- ESTABLISHED

S20G	CONNECTION -
	ESTABLISHING
	-CALLING
		Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FAS error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := FAS error
		},
		reset VCR
			CONNECTION-
			NOT-
			ESTABLISHED

S21G	CONNECTION -
	ESTABLISHING
	-CALLING
		FAS service primitive other than ASC.cnf and ABT.ind
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode :=Connection state conflict FAS
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode :=Connection state conflict FAS
		},
		reset VCR
			CONNECTION-
			NOT-
			ESTABLISHED

S22G	CONNECTION -
	ESTABLISHING
	-CALLING
		ASC.cnf
		&& Data = not allowed, unknown or faulty FMS PDU
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FMS PDU error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := FMS PDU error
		},
		reset VCR
			CONNECTION-
			NOT-
			ESTABLISHED

S23G	CONNECTION -
	ESTABLISHING
	-CALLING
		FMS service.primitive other than Initiate.rsp and Abort.req
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := User error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := User error
		},
		reset VCR
			CONNECTION-
			NOT-
			ESTABLISHED

S24D	CONNECTION -
	ESTABLISHING
	-CALLED
		Initiate.rsp(+)
		=>
		ASC.rsp(+){
		Data = Initiate-ResponsePDU
		}
			CONNECTION-
			ESTABLISHED

S25D	CONNECTION -
	ESTABLISHING
	-CALLED
		Initiate.rsp(-)
		=>
		ASC.rsp(-){
		Data = Initiate-ErrorPDU{
		ErrorCode = ErrorCode of Initiate.rsp
		}
		},
		reset VCR
			CONNECTION-
			NOT-
			ESTABLISHED

#	Current state	Event or condition => action	Next state
S26D	CONNECTION - ESTABLISHING -CALLED	Abort.req => ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED
S27D	CONNECTION - ESTABLISHING -CALLED	ABT.ind => Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED

S28D	CONNECTION - ESTABLISHING -CALLED	Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FAS error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := FAS error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S29D	CONNECTION - ESTABLISHING -CALLED	FAS service primitive other ABT.ind
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode :=Connection state conflict FAS
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode :=Connection state conflict FAS
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S30D	CONNECTION - ESTABLISHING -CALLED	FMS service.primitive other than Initiate.rsp and Abort.req
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := User error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := User error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S31	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) positive => DTC.req{ Data := Confirmed-RequestPDU }, OSCS := OSCS + 1	CONNECTION-ESTABLISHED

S32	CONNECTION-ESTABLISHED	ConfirmedService.req
		&& OSCS = ActualMaxSCC
		=>
		Reject.ind{
		DetectedHere := true,
		Original InvokeID := InvokeID in ConfirmedService.req,
		RejectPDUType := Confirmed-RequestPDU,
		RejectCode := Max-services-overflow
		}
			CONNECTION-ESTABLISHED

#	Current state	Event or condition => action	Next state

S33	CONNECTION-ESTABLISHED	ConfirmedService.req
		&& OSCS < ActualMaxSCC
		&& InvokeID already existent
		=>
		Reject.ind{
		DetectedHere := true,
		Original InvokeID := InvokeID in ConfirmedService.req,
		RejectPDUType := Confirmed-RequestPDU,
		RejectCode := Invoke-ID-exists
		}
			CONNECTION-ESTABLISHED

S34	CONNECTION-ESTABLISHED	ConfirmedService.req
		&& OSCS < ActualMaxSCC
		&& InvokeID not existent
		&& PDU length > Max FMS PDU sending
		=>
		Reject.ind{
		DetectedHere := true,
		Original InvokeID := InvokeID in ConfirmedService.req,
		RejectPDUType := Confirmed-RequestPDU,
		RejectCode := PDU-size
		}
			CONNECTION-ESTABLISHED

S35	CONNECTION-ESTABLISHED	ConfirmedService.req
		&& OSCS < ActualMaxSCC
		&& InvokeID not existent
		&& PDU length ≤ Max FMS PDU sending
		&& FmsFeaturesSupported test (client) negative
		=>
		Reject.ind{
		DetectedHere := true,
		Original InvokeID := InvokeID in ConfirmedService.req,
		RejectPDUType := Confirmed-RequestPDU,
		RejectCode := Feature-not-supported-connection-oriented
		}
			CONNECTION-ESTABLISHED

S36	CONNECTION-ESTABLISHED	UnconfirmedService.req
		&& PDU length ≤ Max FMS PDU sending
		&& FmsFeaturesSupported test (client) positive
		=>
		DTU.req{
		Data := Unconfirmed-PDU
		}
			CONNECTION-ESTABLISHED

S37	CONNECTION-ESTABLISHED	UnconfirmedService.req
		&& PDU length > Max FMS PDU sending
		=>
		Reject.ind{
		DetectedHere := true,
		Original InvokeID := InvokeID in UnconfirmedService.req,
		RejectPDUType := Unconfirmed-PDU,
		RejectCode := PDU-size
		}
			CONNECTION-ESTABLISHED

S38	CONNECTION-ESTABLISHED	UnconfirmedService.req
		&& PDU length ≤ Max FMS PDU sending
		&& FmsFeaturesSupported test (client) negative
		=>
		Reject.ind{
		DetectedHere := true,
		Original InvokeID := InvokeID in UnconfirmedService.req,
		RejectPDUType := Unconfirmed-PDU,
		RejectCode := Feature-not-supported-connection-oriented
		}
			CONNECTION-ESTABLISHED

S39	CONNECTION-ESTABLISHED	Abort.req => ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	**Event** or **condition** **=> action**	Next state

S40	CONNECTION-ESTABLISHED	Faulty, unknown or not-allowed FMS service.primitive
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := User error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := User error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S41	CONNECTION-ESTABLISHED	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR < ActualMaxRCC
		&& InvokeID not existent
		&& Features Supported test (server) positive
		=>
		ConfirmedService.ind{
		}
		OSCR := OSCR + 1
			CONNECTION-ESTABLISHED

S42	CONNECTION-ESTABLISHED	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length > Max FMS PDU receiving
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := PDU-size
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := PDU-size
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S43	CONNECTION-ESTABLISHED	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR ≥ ActualMaxRCC
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := Max-services-overflow
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := Max-services-overflow
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S44	CONNECTION-ESTABLISHED	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR < ActualMaxRCC
		&& InvokeID already existent
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-request
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-request
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state

S45	CONNECTION-ESTABLISHED	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR < ActualMaxRCC
		&& InvokeID not existent
		&& Features Supported test (server) negative
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := Feature-not-supported
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := Feature-not-supported
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S46	CONNECTION-ESTABLISHED	DTU.ind
		&& Data = Unconfirmed-PDU
		&& PDU length ≤ Max FMS PDU receiving
		&& Features Supported test (server) positive
		=>
		UnconfirmedService.ind{
		}
			CONNECTION-ESTABLISHED

S47	CONNECTION-ESTABLISHED	DTU.ind
		&& Data = Unconfirmed-PDU
		&& PDU length > Max FMS PDU receiving
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := PDU-size
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := PDU-size
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S48	CONNECTION-ESTABLISHED	DTU.ind
		&& Data = Unconfirmed-PDU
		&& PDU length ≤ Max FMS PDU receiving
		&& Features Supported test (server) negative
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := Feature-not-supported
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := Feature-not-supported
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S49	CONNECTION-ESTABLISHED	ABT.ind => Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED

S50	CONNECTION-ESTABLISHED	ASC.ind
		&& Data = Initiate-RequestPDU
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := Connection-state-conflict-FMS
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := Connection-state-conflict-FMS
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

#	Current state	**Event** or **condition** **=> action**	Next state

S51	CONNECTION-ESTABLISHED	Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FAS error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := FAS error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S52	CONNECTION-ESTABLISHED	Not-allowed FAS service primitive
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := Connection-state-conflict-FAS
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := Connection-state-conflict-FAS
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S53	CONNECTION-ESTABLISHED	valid FAS service primitive
		&& Data = not-allowed, unknown or faulty FMS PDU
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FMS-PDU-error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := FMS-PDU-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S54	CONNECTION-ESTABLISHED	ConfirmedService.rsp && InvokeID (server) existent && services in .rsp and .ind are identical && PDU length ≤ Max FMS PDU sending => DTC.rsp{ Data := Confirmed-ResponsePDU }, OSCR := OSCR – 1	CONNECTION-ESTABLISHED

S55	CONNECTION-ESTABLISHED	ConfirmedService.rsp
		&& InvokeID (server) not existent
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-response
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-response
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S56	CONNECTION-ESTABLISHED	ConfirmedService.rsp
		&& InvokeID (server) existent
		&& services in .rsp and .ind are not identical
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := service-error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := service-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state

S57	CONNECTION-ESTABLISHED	ConfirmedService.rsp
		&& InvokeID (server) existent
		&& services in .rsp and .ind are identical
		&& PDU length > Max FMS PDU sending
		=>
		DTC.rsp{
		Data := Reject-PDU{
		Original InvokeID := InvokeID in ConfirmedService.req,
		RejectCode := PDU-size
		}
		},
		OSCR := OSCR – 1
			CONNECTION-ESTABLISHED

S58	CONNECTION-ESTABLISHED	DTC.cnf && Data = Confirmed-ResponsePDU && PDU length ≤ Max FMS PDU receiving && InvokeID (client) existent && services in .cnf and .req are identical => ConfirmedService.cnf{ }, OSCR := OSCR – 1	CONNECTION-ESTABLISHED

S59	CONNECTION-ESTABLISHED	DTC.cnf
		&& Data = Confirmed-ResponsePDU
		&& PDU length > Max FMS PDU receiving
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := PDU-size
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := PDU-size
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S60	CONNECTION-ESTABLISHED	DTC.cnf
		&& Data = Confirmed-ResponsePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& InvokeID (client) not existent
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-response
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-response
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S61	CONNECTION-ESTABLISHED	DTC.cnf && Data = Reject-PDU && Original InvokeID (client) existent && RejectCode:= PDU-size => Reject.ind{ DetectedHere := false, Original InvokeID := InvokeID in Reject-PDU, RejectPDUType := Confirmed-Response-PDU, RejectCode := PDU-size }, OSCS := OSCS -1	CONNECTION-ESTABLISHED

S62	CONNECTION-ESTABLISHED	DTC.cnf
		&& Data = Reject-PDU
		&& Original InvokeID (client) not existent
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-response
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := InvokeID-error-response
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state

S63	CONNECTION-ESTABLISHED	DTC.cnf
		&& Data = Reject-PDU
		&& Original InvokeID (client) existent
		&& RejectCode:<> PDU-size
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := FMS-PDU-error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := FMS-PDU-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

S64	CONNECTION-ESTABLISHED	DTC.cnf
		&& Data = Confirmed-ResponsePDU
		&& InvokeID (client) existent
		&& Services in .req and .cnf are not identical
		=>
		ABT.req{
		AbortIdentifier := FMS,
		ReasonCode := Service-error
		},
		Abort.ind{
		AbortIdentifier := FMS,
		ReasonCode := Service-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED

7.2.4 Primitives exchanged between FAL-User and VCR PM

See Table 4 and Table 5 for the primitives exchanged between the FAL-User and the VCR PM.

Table 4 – Primitives issued by FAL-User to VCR PM

Primitive name	Source	Associated parameters and functions
Terminate.req	FAL-User	Refer to FAL service Definition (see IEC 61158-5-9)
Initiate.req	FAL-User	
Initiate.rsp(+)	FAL-User	
Initiate.rsp(-)	FAL-User	
ConfirmedService.req	FAL-User	
ConfirmedService.rsp	FAL-User	
UnconfirmedService.req	FAL-User	

Table 5 – Primitives issued by VCR PM to FAL-User

Primitive name	Source	Associated parameters and functions
Terminate.ind	VCR PM	Refer to FAL service Definition (IEC 61158-5-9)
Initiate.ind	VCR PM	
Initiate.cnf(+)	VCR PM	
Initiate.cnf(-)	VCR PM	
ConfirmedService.ind	VCR PM	
ConfirmedService.cnf	VCR PM	
UnconfirmedService.ind	VCR PM	

7.2.5 Primitives exchanged between FSPM and the VCR PM

Table 6 and Table 7 define the primitives used by the FSPM.

Table 6 – Primitives issued by VCR PM to FSPM

Primitive name	Source	Associated parameters	Functions
ASC.req	VCR PM	Arep_Id, Data, Remote_DLCEP_Address	This primitive is used to convey an Associate request primitive from the VCR PM to the FSPM.
ASC.rsp(+)	VCR PM	Arep_Id, Data	This primitive is used to convey an Associate response(+) primitive from the VCR PM to the FSPM.
ASC.rsp(-)	VCR PM	Arep_Id, Data	This primitive is used to convey an Associate response(-) primitive from the VCR PM to the FSPM.
Abort.req	VCR PM	Arep_Id, Identifier, Reason_Code, Additional_Detail	This primitive is used to convey an Abort request primitive from the VCR PM to the FSPM.
CS.req	VCR PM	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) request primitive from the VCR PM to the FSPM.
CS.rsp	VCR PM	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) response primitive from the VCR PM to the FSPM.
UCS.req	VCR PM	Arep_Id, Remote_DLSAP_Address, Data	This primitive is used to convey an Unconfirmed Send (UCS) request primitive from the VCR PM to the FSPM.
FCMP.req	VCR PM	Arep_Id	This primitive is used to convey an FAL-Compel (FCMP) request primitive from the VCR PM to the FSPM.
GBM.req	VCR PM	Arep_Id	This primitive is used to convey a Get-Buffered-Message (GBM) request primitive from the VCR PM to the FSPM.

Table 7 – Primitives issued by FSPM to VCR PM

Primitive name	Source	Associated parameters	Functions
ASC.ind	FSPM	Arep_Id, Data	This primitive is used to convey an Associate indication primitive from the FSPM to the VCR PM.
ASC.cnf(+)	FSPM	Arep_Id, Data	This primitive is used to convey an Associate result(+) primitive from the FSPM to the VCR PM.
ASC.cnf(-)	FSPM	Arep_Id, Data	This primitive is used to convey an Associate result(-) primitive from the FSPM to the VCR PM.
Abort.ind	FSPM	Arep_Id, Locally_Generated, Identifier, Reason_Code, Additional_Detail	This primitive is used to convey an Abort indication primitive from the FSPM to the VCR PM.
CS.ind	FSPM	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) indication primitive from the FSPM to the VCR PM.
CS.cnf	FSPM	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) confirmation primitive from the FSPM to the VCR PM.
UCS.ind	FSPM	Arep_Id, Remote_DLSAP_Address, Duplicate_FAL-SDU, Data, Local_Timeliness, Remote_Timeliness	This primitive is used to convey an Unconfirmed Send (UCS) indication primitive from the FSPM to the VCR PM.

Primitive name	Source	Associated parameters	Functions
FCMP.cnf	FSPM	Arep_Id, Status	This primitive is used to convey an FAL-Compel (FCMP) confirmation primitive from the FSPM to the VCR PM.
GBM.cnf(+)	FSPM	Arep_Id, Duplicate_FAL-SDU, Data, Local_Timeliness, Remote_Timeliness	This primitive is used to convey a Get-Buffered-Message (GBM) positive confirmation primitive from the FSPM to the VCR PM.
GBM.cnf(-)	FSPM	Arep_Id	This primitive is used to convey a Get-Buffered-Message (GBM) negative confirmation primitive from the FSPM to the VCR PM.
FSTS.ind	FSPM	Arep_Id, Reported_Status	This primitive is used to convey an FAL-Status (FSTS) indication primitive from the FSPM to the VCR PM.

8 FAL service protocol machine (FSPM)

8.1 General

FAS service Protocol Machine is common to all the AREP types. Only applicable primitives are different among different AREP types. It has one state called "ACTIVE" as shown in Figure 23.

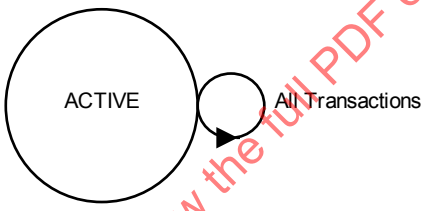


Figure 23 – State transition diagram of FSPM

8.2 FSPM state tables

Table 8 and Table 9 specifiy the FSPM protocol machine.

Table 8 – FSPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	ASC.req
		&& SelectArep (Arep_Id) = "True"
		=>
		ASC_req {
		user_data := Data,
		remote_dlcep_address := Remote_DLCEP_Address
		}
			ACTIVE

S2	ACTIVE	ASC.rsp(+)
		&& SelectArep (Arep_Id) = "True"
		=>
		ASC_rsp(+) {
		user_data := Data
		}
			ACTIVE

S3	ACTIVE	ASC.rsp(-)
		&& SelectArep (Arep_Id) = "True"
		=>
		ASC_rsp(-) {
		user_data := Data
		}
			ACTIVE

S4	ACTIVE	DTU.req
		&& SelectArep (Arep_Id) = "True"
		=>
		DTU_req {
		remote_dlsap_address := Remote_DLSAP_Address,
		user_data := Data
		}
			ACTIVE

S5	ACTIVE	DTC.req
		&& SelectArep (Arep_Id) = "True"
		=>
		DTC_req {
		user_data := Data
		}
			ACTIVE

S6	ACTIVE	DTC.rsp
		&& SelectArep (Arep_Id) = "True"
		=>
		DTC_rsp {
		user_data := Data
		}
			ACTIVE

S7	ACTIVE	Abort.req
		&& SelectArep (Arep_Id) = "True"
		=>
		Abort_req {
		Identifier := Identifier,
		reason_code := Reason_Code,
		additional_detail := Additional_Detail
		}
			ACTIVE

S8	ACTIVE	FCMP.req
		&& SelectArep (Arep_Id) = “True”
		=>
		FCMP_req {
		}
			ACTIVE

S9	ACTIVE	GBM.req
		&& SelectArep (Arep_Id) = “True”
		=>
		GBM_req {
		}
			ACTIVE

NOTE 1 A primitive generated in the FSPM sender state machine is sent to an appropriate ARPM that is selected by the FSPM using the SelectArep function. The Arep_Id parameter supplied by the FAS user is the argument of this function.

NOTE 2 If the SelectArep function returns the value of False, it is a local matter to report such instance and the FSPM does not generate any primitives to the ARPM.

Table 9 – FSPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	ACTIVE	ASC_ind
		=>
		ASC.ind {
		Arep_Id := arep_id,
		Data := user_data
		}
			ACTIVE

R2	ACTIVE	ASC_cnf(+)
		=>
		ASC.cnf(+) {
		Arep_Id := arep_id,
		Data := user_data
		}
			ACTIVE

R3	ACTIVE	ASC_cnf(-)
		=>
		ASC.cnf(-) {
		Arep_Id := arep_id,
		Data := user_data
		}
			ACTIVE

R4	ACTIVE	DTU_ind
		=>
		DTU.ind {
		Arep_Id := arep_id,
		Remote_DLSAP_Address := remote_dlsap_address,
		Duplicate_FAS-SDU := duplicate_fas_sdu,
		Data := user_data,
		Local_Timeliness := local_timeliness,
		Remote_Timeliness := remote_timeliness
		}
			ACTIVE

R5	ACTIVE	DTC_ind
		=>
		DTC.ind {
		Arep_Id := arep_id,
		Data := user_data
		}
			ACTIVE

R6	ACTIVE	DTC_cnf
		=>
		DTC.cnf {
		Arep_Id := arep_id,
		Data := user_data
		}
			ACTIVE

R7	ACTIVE	Abort_ind
		=>
		Abort.ind {
		Arep_Id := arep_id,
		Locally_Generated := locally_generated,
		Identifier := identifier,
		Reason_Code := reason_code,
		Additional_Detail := additional_detail
		}
			ACTIVE

R8	ACTIVE	FCMP_cnf
		=>
		FCMP.cnf {
		Arep_Id := arep_id,
		Status := status
		}
			ACTIVE

R9	ACTIVE	GBM_cnf(+)
		=>
		GBM.cnf(+) {
		Arep_Id := arep_id,
		Duplicate_FAS-SDU := duplicate_fas_sdu,
		Data := user_data,
		Local_Timeliness := local_timeliness,
		Remote_Timeliness := remote_timeliness
		}
			ACTIVE

R10	ACTIVE	GBM_cnf(-)
		=>
		GBM.cnf(-) {
		Arep_Id := arep_id,
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

R11	ACTIVE	FSTS_ind
		=>
		FSTS.ind {
		Arep_Id := arep_id,
		Reported_Status := reported_status
		}
			ACTIVE

8.3 Functions used by FSPM

Table 10 defines the function used by the FSPM.

Table 10 – Function SelectArep()

Name	SelectArep	Used in	FSPM
Input		Output	
Arep_Id		True || False	
Function			
Looks for the AREP entry that is specified by the Arep_Id parameter. The Arep_Id parameter is provided with the FAS user service primitives.			

8.4 Parameters of FSPM/ARPM primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in Table 11.

Table 11 – Parameters used with primitives exchanged between FSPM and ARPM

Parameter name	Description
arep_id	This parameter is used to unanimously identify an instance of the AREP that has issued a primitive. A means for such identification is not specified by this specification.
user_data	This parameter conveys FAS-User data.
locally_generated	This parameter conveys value that is used for the Locally_Generated parameter.
identifier	This parameter conveys value that is used for the Identifier parameter.
reason_code	This parameter conveys value that is used for the Reason_Code parameter.
additional_detail	This parameter conveys value that is used for the Additional_Detail parameter.
duplicate_fas_sdu	This parameter conveys value that is used for the Duplicate_FAS-SDU parameter.
remote_dlsap_address	This parameter conveys value that is used for the Remote_DLSAP_Address parameter.
status	This parameter conveys value that is used for the Status parameter.
reported_status	This parameter conveys a Data Like Layer event status.
local_timeliness	This parameter conveys value that is used for the Local_Timeliness parameter.
remote_timeliness	This parameter conveys value that is used for the Remote_Timeliness parameter.

9 Application relationship protocol machines (ARPMs)

9.1 AREP mapping to data-link layer

9.1.1 General

Subclause 9.1 describes the mapping of the FAS to the Fieldbus data-link layer. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the data-link layer specification; rather, it defines how they are used by each of the AR classes. A means to configure and monitor the values of these attributes is provided by Network Management.

The following class definitions describe the DLSAP attributes and the DLME attributes required to support each of the AREP classes.

NOTE Undefined attributes use the same definitions as those previously defined.

9.1.2 DLL mapping of QUU AREP class

9.1.2.1 QUU AREP class formal model

The DLL Mapping attributes and their permitted values and the DLL services used with the QUU AREP class are defined in this subclause.

CLASS: **QuuCl**
PARENT CLASS: QueuedUser-TriggeredUnidirectionalAREP
ATTRIBUTES:

1.	(m)	KeyAttribute:	LocalDlsapAddress
2.	(m)	Attribute:	RemoteDlsapAddress
3.	(m)	Attribute:	LocalDlsapRole (Basic, Group)
4.	(c)	Constraint:	Role = Source
4.1	(m)	Attribute:	DefaultQosAsSender
4.2	(m)	Attribute	DllPriority (Urgent, Normal, TimeAvailable)
4.3	(m)	Attribute:	MaxConfirmDelayOnUnitdata
4.4	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
4.5	(m)	Attribute:	DlSchedulingPolicy (Implicit)
4.6	(m)	Attribute:	ExplicitQueue (True, False)
5.	(c)	Constraint:	ExplicitQueue = True
5.1	(m)	Attribute:	QueueBindings—either for a sender or a receiver
5.2	(m)	Attribute:	QueueIdentifier
5.3	(m)	Attribute:	MaxQueueDepth
5.4	(m)	Attribute:	MaxDlsduSize

DLL SERVICES:

1.	(m)	OpsService:	DL-Unitdata
2.	(c)	Constraint:	ExplicitQueue = True
3.	(m)	OpsService:	DL-Get

9.1.2.2 Attributes

9.1.2.2.1 LocalDlsapAddress

This attribute specifies the DLSAP address to which this AREP is attached. This attribute is a DLSAP-address if the Role attribute has the value of Source, and either a group DL-address or a DLSAP-address if the Role attribute has the value of Sink.

This attribute supplies the value for the “DL(SAP)-address” parameter specified in the DLL.

This attribute contains the following three sub-attributes: Link Address, Node Address, and Selector.

9.1.2.2.2 RemoteDlsapAddress

This attribute specifies the remote address to which FAS-PDUs are sent (for Source AREPs), or from which they are received (for Sink AREPs).

If the ConfigurationType attribute is Linked, the value of this attribute has been configured. If it is Free, the value of this attribute is provided with a service request.

This attribute contains the following three sub-attributes: Link Address, Node Address, and Selector.

9.1.2.2.3 LocalDlsapRole

This attribute specifies the behavior of the local DLSAP to be used. If the Role is Source, this attribute has the value of Basic. If the Role is Sink, it has the value of either Basic or Group.

This attribute supplies the value for the “DL(SAP)-role” parameter specified by the DLL.

9.1.2.2.4 DefaultQosAsSender

The DefaultQosAsSender attributes specify the DLL quality of service that is used by the sending AREP. The receiving DLL shall support the quality of service specified by these attributes.

9.1.2.2.5 DllPriority

This attribute defines the DLL priority, and thus restricts the maximum length of an FAS-PDU, of the conveyance path of an AR.

This attribute supplies the value for the “DLL priority” parameter of the DLL. The values Urgent, Normal, and Time-Available correspond to URGENT, NORMAL, and TIME-AVAILABLE as defined in IEC 61158-3-1 and IEC 61158-4-1, respectively.

NOTE It is not possible to use different priorities for each FAS-PDU sent from the same QUU AREP.

9.1.2.2.6 MaxConfirmDelayOnUnitdata

This attribute specifies the maximum confirmation delay for a local confirmation from a DL-Unitdata request primitive.

This attribute supplies the value for the “Max confirm delay on locally-confirmed DL-Unitdata” parameter of the DLL.

9.1.2.2.7 DlpduAuthentication

This attribute specifies the lower bound of the length of the DL-addresses to be used by the DLL.

This attribute supplies the value for the “DLPDU authentication” parameter of the DLL. The values Ordinary, Source, and Maximal correspond to ORDINARY, SOURCE, and MAXIMAL as defined in IEC 61158-3-1 and IEC 61158-4-1, respectively.

9.1.2.2.8 DlSchedulingPolicy

This attribute provides the guidance to the DLL on the scheduling needed by an AR. For this AREP, the DLL tries to transmit the FAS-PDU as soon as it is passed from the FAS.

This attribute supplies the value for DL-Scheduling-policy attribute. Only the value Implicit is used. It corresponds to IMPLICIT as defined in IEC 61158-3-1 and IEC 61158-4-1.

9.1.2.2.9 ExplicitQueue

This attribute specifies, when True, that the characteristics of the associated sending and receiving queues are explicitly configured and managed through the Network Management. The value of False means that queues with implementation specific depth and length are provided by the DLL.

9.1.2.2.10 QueueBindings

The following attributes specify the explicit queue that is bound to this DLSAP. For a sender, the queue is for sending. For a receiver, it is for receiving.

9.1.2.2.11 QueueIdentifier

This attribute provides a local means to identify a queue that is associated with this DLSAP.

This attribute supplies the value for the “Buffer-or-queue-identifier” parameter defined in the DLL.

9.1.2.2.12 MaxQueueDepth

This attribute specifies the maximum number of FAS-PDUs that can be queued for sending or receiving.

This attribute supplies the value for the “Maximum queue depth” parameter of the DLL.

9.1.2.2.13 MaxDlsduSize

This attribute specifies the maximum length of an FAS-PDU that can be sent or received by the DLL.

This attribute supplies the value for the “Maximum DLSDU size” parameter of the DLL.

9.1.2.3 DLL services

Refer to IEC 61158-3-1 for DLL service descriptions.

9.1.3 DLL mapping of QUB AREP class

9.1.3.1 QUB AREP class formal model

The DLL Mapping attributes and their permitted values and the DLL services used with the QUB AREP class are defined in this subclause.

CLASS: **QubCo**
PARENT CLASS: **QueuedUser-TriggeredBidirectionalAREP**
ATTRIBUTES:

1.	(m)	KeyAttribute:	LocalDlcepAddress
2.	(m)	Attribute:	RemoteDlcepAddress
3.	(m)	KeyAttribute:	DlcepDlIdentifier
4.	(m)	Attribute:	DlsapRole (Basic)
5.	(m)	Attribute:	QosParameterSet
5.1	(m)	Attribute:	DlcepClass (Peer)
5.2	(m)	Attribute:	DlcepDataDeliveryFeatures
5.2.1	(m)	Attribute:	FromRequesterToResponder (Classical, Disordered)
5.2.2	(m)	Attribute:	FromResponderToRequester (Classical, Disordered)
5.3	(m)	Attribute:	Priority
5.3.1	(m)	Attribute:	DllPriority (Urgent, Normal, TimeAvailable)
5.3.2	(m)	Attribute:	DllPriorityNegotiated (Urgent, Normal, TimeAvailable)
5.4	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
5.5	(m)	Attribute:	ResidualActivity
5.5.1	(m)	Attribute:	ResidualActivityAsSender (True, False)
5.5.2	(m)	Attribute:	ResidualActivityAsReceiver (True, False)
5.6	(m)	Attribute:	MaxConfirmDelay
5.6.1	(m)	Attribute:	MaxConfirmDelayOnDlConnect
5.6.2	(m)	Attribute:	MaxConfirmDelayOnDlData
5.7	(m)	Attribute:	DlSchedulingPolicy (Implicit)
5.8	(m)	Attribute:	ExplicitQueue (True, False)
5.9	(c)	Constraint:	ExplicitQueue = True
5.9.1	(m)	Attribute:	MaxDlsduSizes
5.9.2	(m)	Attribute:	MaxDlsduSizeFromRequester
5.9.3	(m)	Attribute:	MaxDlsduSizeFromResponder
5.9.4	(m)	Attribute:	MaxDlsduSizeFromRequesterNegotiated
5.9.5	(m)	Attribute:	MaxDlsduSizeFromResponderNegotiated
5.10	(m)	Attribute:	QueueBindings
5.10.1	(m)	Attribute:	SendingBufferOrQueueIdentifier
5.10.2	(m)	Attribute:	ReceivingBufferOrQueueIdentifier
5.11	(m)	Attribute:	MaxQueueDepth
5.11.1	(m)	Attribute:	MaxSendingQueueDepth
5.11.2	(m)	Attribute:	MaximimReceivingQueueDepth

DLL SERVICES:

1.	(m)	OpsService:	DL-Data
2.	(c)	Constraint:	ExplicitQueue = True
2.1	(m)	OpsService:	DL-Get
3.	(m)	OpsService:	DL-Connect
4.	(m)	OpsService:	DL-Connection-Established
5.	(m)	OpsService:	DL-Disconnect

9.1.3.2 Attributes

9.1.3.2.1 LocalDlcepAddress

This attribute specifies the local DLCEP address and identifies the DLCEP.

The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

NOTE 1 The value of this attribute is also carried in the Associate Request PDU.

This attribute contains the following three sub-attributes: Link Address, Node Address, and Selector.

NOTE 2 Since the local Link and Node addresses are set by the Network Management, only the Selector portion of the LocalDlsapAddress attribute is a configurable attribute of the FAS.

9.1.3.2.2 RemoteDlcepAddress

This attribute specifies the remote DLCEP address and identifies the DLCEP.

This attribute supplies the value for called DLCEP-address of the DL-Connect service.

NOTE The value of this attribute is also carried in the header part of the Associate Request PDU.

This attribute contains the following three sub-attributes: Link Address, Node Address, and Selector.

9.1.3.2.3 DlcepDlIdentifier

The DlcepDlIdentifier attribute identifies the DLCEP.

This attribute supplies the value for the DLCEP DL-identifier parameter of the DL-Connect service. It is returned to the calling FAS by the DL-Connect request primitive, and by the DL-Connect indication primitive to the called FAS.

9.1.3.2.4 DlsapRole

This attribute specifies the behavior of the DLSAP that is used by the AREP.

This attribute supplies the value for the “DL(SAP)-role” parameter. The possible value Basic corresponds to BASIC as defined in the data-link layer specification.

9.1.3.2.5 QosParameterSet

The QosParameterSet attributes specify the DL quality of service that is used by this AREP. These attribute values may be negotiated with the remote AREP.

9.1.3.2.6 DlcepClass

This attribute specifies the behavior of the DLCEP which is attached to the AREP.

This attribute supplies the value for the “DLCEP class” parameter of the DLL. The possible value of this attribute is Peer and corresponds to Peer defined by the DLL.

9.1.3.2.7 DlcepDataDeliveryFeatures

These two attributes specify data delivery features of the DLL.

The permitted values Classical and Disordered correspond, respectively to CLASSICAL and DISORDERED defined by the data-link layer specification.

The FromRequesterToResponder and FromResponderToRequester attributes shall have the same value.

9.1.3.2.8 FromRequesterToResponder

This attribute specifies the DLL data delivery feature of the DLPDUs sent from the AREP whose Initiator attribute has a value of “True” to the remote AREP. It supplies the value for the “DLCEP data delivery features from requester to responder(s)” parameter defined in the DLL.

9.1.3.2.9 FromResponderToRequester

This attribute specifies the DLL data delivery feature of the DLPDUs sent from the AREP whose Initiator attribute has a value of “False” to the remote AREP. It supplies the value for the “DLCEP data delivery features from responder(s) to requester” parameter defined in the DLL.

9.1.3.2.10 DllPriority

This attribute specifies the configured value of the DLL priority.

It is not possible to use different priorities for each FAS-PDU sent from the same QUB AREP. Also, it is not permitted to use different priorities for the send and receive conveyance paths.

9.1.3.2.11 DllPriorityNegotiated

This attribute specifies the negotiated value of the DLL priority.

9.1.3.2.12 DlpduAuthentication

This attribute specifies the lower bound of the length of DL-addressing to be used by the DLL.

This attribute supplies the value for the “DLPDU-authentication” parameter of the DLL. The permitted value Ordinary, Source, and Maximal correspond to ORDINARY, Source, and MAXIMAL, respectively, as defined in IEC 61158-3-1 and IEC 61158-4-1. ResidualActivityAsSender

This attribute specifies sender’s DLC residual activity. It supplies the value for the “Residual activity as sender” parameter defined in the DLL. The possible values are “True” and “False.”

9.1.3.2.13 ResidualActivityAsReceiver

This attribute specifies receiver’s DLC residual activity. It supplies the value for the “Residual activity as receiver” parameter defined in the DLL. The possible values are “True” and “False.”

9.1.3.2.14 MaxConfirmDelay

This attribute specifies the maximum confirmation delay of certain DLL connection-oriented services.

9.1.3.2.15 MaxConfirmDelayOnDlConnect

This attribute species the maximum confirmation delay for a confirmation from a DL-Connect service.

This attribute supplies the value for the “Maximum confirmation delay on DL-Connect, DL-Reset and DL-Subscriber-Query” parameter specified in the data-link layer specification.

9.1.3.2.16 MaxConfirmDelayOnDlData

This attribute species the maximum confirmation delay for a confirmation from a DL-Data service.

This attribute supplies the value for the “Maximum confirmation delay on DL-Data” parameter specified in IEC 61158-3-1 and IEC 61158-4-1.

9.1.3.2.17 DlSchedulingPolicy

This attribute provides the guidance to the DLL on the scheduling needed by an AR. For this AREP, the DLL tries to transmit the FAS-PDU as soon as possible.

This attribute supplies the value for the “DL-Scheduling-policy” parameter of the DLL. The permitted value Implicit corresponds to IMPLICIT defined in IEC 61158-3-1 and IEC 61158-4-1.

9.1.3.2.18 ExplicitQueue

9.1.3.2.19 MaxDlsduSizeFromRequester

This attribute specifies the configured value of the maximum length of an FAS-PDU that can be sent from the AREP whose Initiator attribute has a value of “True” to the remote AREP.

This attribute supplies the value for the “Maximum DLSDU sizes from requester” parameter of the DLL.

9.1.3.2.20 MaxDlsduSizeFromResponder

This attribute specifies the configured value of the maximum length of an FAS-PDU that can be sent from the AREP whose Initiator attribute has a value of “False” to the remote AREP.

This attribute supplies the value for the “Maximum DLSDU sizes from responder” parameter of the DLL.

9.1.3.2.21 MaxDlsduSizeFromRequesterNegotiated

This attribute specifies the negotiated value of the maximum length of an FAS-PDU that can be sent from the AREP whose Initiator attribute has a value of “True” to the remote AREP.

9.1.3.2.22 MaxDlsduSizeFromResponderNegotiated

This attribute specifies the negotiated value of the maximum length of an FAS-PDU that can be sent from the AREP whose Initiator attribute has a value of “False” to the remote AREP.

9.1.3.2.23 SendingBufferOrQueueIdentifier

This attribute provides a local means to identify a queue that is used to store sending FAS-PDUs.

This attribute supplies the value for the “Buffer-and-queue bindings as sender” parameter of the DLL.

9.1.3.2.24 ReceivingBufferOrQueueIdentifier

This attribute provides a local means to identify a queue that is used to store receiving FAS-PDUs.

This attribute supplies the value for the “Buffer-and-queue bindings as receiver” parameter of the DLL.

9.1.3.2.25 MaxSendingQueueDepth

This attribute specifies the maximum number of FAS-PDUs that can be queued for transmission.

This attribute supplies the value for the “Maximum queue depth” parameter of the DLL for Send queues.

9.1.3.2.26 MaxReceivingQueueDepth

This attribute specifies the maximum number of FAS-PDUs that can be queued at reception.

This attribute supplies the value for the “Maximum queue depth” parameter of the DLL for Receive queues.

9.1.3.3 DLL services

Refer to IEC 61158-3-1 for DLL service descriptions.

9.1.4 DLL mapping of BNU AREP class

9.1.4.1 BNU AREP class formal model

The DLL Mapping attributes and their permitted values and the DLL services used with the BNU AREP class are defined in this subclause.

CLASS: **BnuCo**
PARENT CLASS: **BufferedNetworkScheduledUnidirectionalAREP**
ATTRIBUTES:

1.	(m)	KeyAttribute:	PublisherDlcepAddress
2.	(m)	KeyAttribute:	DlcepDlIdentifier
3.	(m)	Attribute:	DlsapRole (Basic)
4.	(m)	Attribute:	QosParameterSet
4.1	(m)	Attribute:	DlcepClass (Publisher, Subscriber)
4.2	(m)	Attribute:	DllPriority (Urgent, Normal, TimeAvailable)
4.3	(m)	Attribute:	DlpduAuthentication (Ordinary, Source)
4.4	(m)	Attribute:	ResidualActivity
4.4.1	(m)	Attribute:	AsSender (False)
4.5	(m)	Attribute:	MaxConfirmDelayOnDlConnect
4.6	(m)	Attribute:	DlSchedulingPolicy (Explicit)
4.7	(m)	Constraint:	DlcepClass = Publisher
4.7.1	(m)	Attribute:	FromRequesterToResponder (Ordered, Unordered)
4.7.2	(m)	Attribute:	MaxDlsduSizeFromRequester
4.7.3	(m)	Attribute:	SendingBufferOrQueueIdentifier
4.7.4	(o)	Attribute:	SenderTimeliness
4.7.4.1	(o)	Attribute:	PublisherDlTimelinessClass (Residence, Update, Synchronous, Transparent, None)
4.7.4.2	(o)	Attribute:	PublisherTimeWindowSize
4.7.4.3	(o)	Constraint:	PublisherDlTimelinessClass == Update || Synchronous
4.7.4.3.1	(o)	Attribute:	PublisherSynchronizingDlcep
4.8	(m)	Constraint:	DlcepClass = Subscriber
4.8.1	(m)	Attribute:	FromResponderToRequester (Ordered, Unordered)

4.8.2	(m)	Attribute:	MaxDlsduSizeFromResponder
4.8.3	(m)	Attribute	ReceivingBufferOrQueueIdentifier
4.8.4	(o)	Attribute:	ReceiverTimeliness
4.8.4.1	(o)	Attribute:	PublisherDlTimelinessClass (Residence, Update, Synchronous, Transparent, None)
4.8.4.2	(o)	Attribute:	PublisherTimeWindowSize
4.8.4.3	(o)	Attribute:	SubscriberDlTimelinessClass (Residence, Update, Synchronous, Transparent, None)
4.8.4.4	(o)	Attribute:	SubscriberTimeWindowSize
4.8.4.5	(o)	Constraint:	SubscriberDlTimelinessClass == Update || Synchronous
4.8.4.5.1	(o)	Attribute:	SubscriberSynchronizingDlcep
5.	(m)	Attribute:	LasScheduled (True, False)

DLL SERVICES:

1.	(m)	OpsService:	DL-Put
2.	(m)	OpsService:	DL-Get
3.	(m)	OpsService:	DL-Connect
4.	(m)	OpsService:	DL-Connection-Established
5.	(m)	OpsService:	DL-Disconnect
6.	(m)	OpsService:	DL-Buffer-Received
7.	(m)	OpsService:	DL-Buffer-Sent
8.	(m)	Constraint:	LasScheduled == False
9.	(m)	OpsService:	DL-Compel-Service

9.1.4.2 Attributes

9.1.4.2.1 PublisherDlcepAddress

This attribute specifies the Publisher's DLCEP address and identifies the DLCEP.

The value of this attribute is used as the "DLCEP-address" parameter of the DLL.

This attribute contains the following three sub-attributes: Link Address, Node Address, and Selector.

NOTE Since the local Link and Node addresses are set by the Network Management, only the Selector portion of the LocalDlsapAddress attribute is a configurable attribute of the FAS.

9.1.4.2.2 DlcepDlIdentifier

9.1.4.2.3 DlsapRole

9.1.4.2.4 QosParameterSet

9.1.4.2.5 DlcepClass

9.1.4.2.6 DllPriority

NOTE It is not possible to use different priorities for each FAS-PDU sent from the same BNU AREP.

9.1.4.2.7 DlpduAuthentication

9.1.4.2.8 ResidualActivity

9.1.4.2.9 MaxConfirmDelayOnDlConnect

9.1.4.2.10 DlSchedulingPolicy

This attribute provides the guidance to the DLL on the scheduling needed by an AR. For this AREP, the DLL tries to transmit the FAS-PDU when it is instructed to do so by a stimulus from the network.

This attribute supplies the value for the "DL-Scheduling-policy" parameter of the DLL. The permitted value Explicit corresponds to EXPLICIT defined in IEC 61158-3-1 and IEC 61158-4-1.

9.1.4.2.11 FromRequesterToResponder

This attribute is used if the Role attribute has a value of “Publisher” and specifies the DLL data delivery feature of the AREP.

It supplies the value for the “DLCEP data delivery features from requester to responder(s)” parameter of the DLL. The possible values Ordered and Unordered correspond to ORDERED and UNORDERED defined in IEC 61158-3-1 and IEC 61158-4-1.

If the DuplicatePduDetectionSupported attribute is True, this attribute has the value of “Ordered.”

9.1.4.2.12 MaxDlsduSizeFromRequester

This attribute is used if the Role attribute has a value of “Publisher” and specifies the maximum length of an FAS-PDU that can be sent from this AREP.

This attribute supplies the value for the “Maximum DLSDU sizes from requester” parameter of the DLL.

9.1.4.2.13 SendingBufferOrQueueIdentifier

This attribute provides a local means to identify a buffer that is used to store FAS-PDUs for sending. This attribute supplies the value for the “Buffer-and-queue bindings as sender” parameter of the DLL.

9.1.4.2.14 FromResponderToRequester

This attribute is used if the Role attribute has a value of “Subscriber” and specifies the DLL data delivery feature of the AREP.

It supplies the value for the “DLCEP data delivery features from responder(s) to requester” parameter of the DLL. The possible values Ordered and Unordered correspond to ORDERED and UNORDERED defined in the data-link layer specification.

If the DuplicatePduDetectionSupported attribute is True, this attribute has the value of “Ordered.”

9.1.4.2.15 MaxDlsduSizeFromResponder

This attribute is used if the Role attribute has a value of “Subscriber” and specifies the maximum length of an FAS-PDU that can be received by this AREP.

This attribute supplies the value for the “Maximum DLSDU size from responder” parameter of the DLL.

9.1.4.2.16 ReceivingBufferOrQueueIdentifier

This attribute provides a local means to identify a buffer that is used to store FAS-PDUs for receiving. This attribute supplies the value for the “Buffer-and-queue bindings as receiver” parameter of the DLL.

9.1.4.2.17 SenderTimeliness

9.1.4.2.18 PublisherDlTimelinessClass

This optional attribute provides the timeliness class of a Publisher provided by the DLL. This attribute supplies the value for the “DL-timeliness-class” parameter of the DLL. The permitted

values Residence, Update, Synchronous, Transparent, and None correspond to RESIDENCE, UPDATE, SYNCHRONOUS, TRANSPARENT, and NONE defined by the DLL.

9.1.4.2.19 PublisherTimeWindowSize

This optional attribute provides time window of a Publisher provided by the DLL. This attribute supplies the value for the “Time window size” parameter of the DLL.

9.1.4.2.20 PublisherSynchronizingDlcep

This optional attribute is present when the PublisherDlTimelinessClass attribute has the value of Update or Synchronous and provides a DLCEP that is used to generate synchronizing events by the DLL. The FAS user may derive the AREP from which the synchronizing events are delivered by this attribute value. This attribute supplies the value for the “synchronizing DLCEP” parameter of the DLL.

9.1.4.2.21 ReceiverTimeliness

9.1.4.2.22 SubscriberDlTimelinessClass

This optional attribute provides the timeliness class of a Subscriber provided by the DLL. This attribute supplies the value for the “DL-timeliness-class” parameter of the DLL. The permitted values Residence, Update, Synchronous, Transparent, and None correspond to RESIDENCE, UPDATE, SYNCHRONOUS, TRANSPARENT, and NONE defined by the DLL.

9.1.4.2.23 SubscriberTimeWindowSize

This optional attribute provides time window of a Subscriber provided by the DLL. This attribute supplies the value for the “Time window size” parameter of the DLL.

9.1.4.2.24 SubscriberSynchronizingDlcep

This optional attribute is present when the SubscriberDlTimelinessClass attribute has the value of Update or Synchronous and provides a DLCEP that is used to generate synchronizing events by the DLL. The FAS user may derive the AREP from which the synchronizing events are delivered by this attribute value. This attribute supplies the value for the “synchronizing DLCEP” parameter of the DLL.

9.1.4.2.25 LasScheduled

This attribute specifies whether this AREP is LAS-scheduled or not. The value of True means that this APRE is LAS-Scheduled. The value of False means that it is not LAS-Scheduled.

NOTE This attribute does not affect the operation of the FAS and the DLL. It is used by the FAS users only.

9.1.4.3 DLL services

Refer to IEC 61158-3-1 and IEC 61158-4-1 for DLL service descriptions.

9.2 Application relationship protocol machines (ARPMs)

9.2.1 AR protocol machine (ARPM) for QUU AREP

9.2.1.1 QUU ARPM states

The defined states and their descriptions of the QUU ARPM are shown in Table 12 and Figure 24.

Table 12 – QUU ARPM states

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

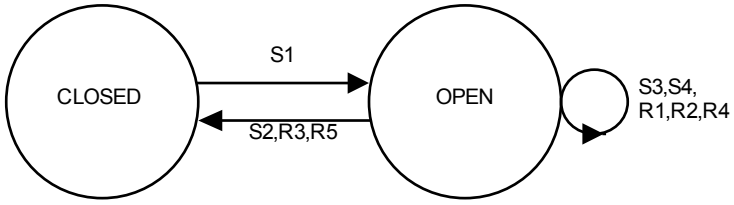


Figure 24 – State transition diagram of the QUU ARPM

9.2.1.2 QUU ARPM state table

Table 13 and Table 14 specify the QUU QRPM state machine.

Table 13 – QUU ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1	CLOSED	ASC_req
		=>
		ASC_cnf(+) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			OPEN

S2	OPEN	Abort_req => (no actions taken)	CLOSED

S3	OPEN	DTU_req
		&& ConfigurationType = “Linked”
		&& Role = “Source”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "DTU_PDU",
		fas_data := user_data)
		}
			OPEN

S4	OPEN	DTU_req
		&& ConfigurationType = “Free”
		&& Role = “Source”
		=>
		RemoteDlsapAddress := remote_dlsap_address,
		
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "DTU_PDU",
		fas_data := user_data)
		}
			OPEN

Table 14 – QUU ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Unitdata_ind”
		&& ConfigurationType = “Linked”
		&& RemoteDlsapAddress = calling_address
		&& Role = “Sink”
		&& FAS_Pdu_Type (fas_pdu) = “DTU_PDU”
		=>
		DTU_ind {
		arep_id := GetArepId (),
		user_data := fas_pdu
		}
			OPEN

R2	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Unitdata_ind”
		&& ConfigurationType = “Free”
		&& Role = “Sink”
		&& FAS_Pdu_Type (fas_pdu) = “DTU_PDU”
		=>
		DTU_ind {
		arep_id := GetArepId (),
		remote_dlsap_address := calling_address,
		user_data := fas_pdu
		}
			OPEN

R3	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Unitdata_ind”
		&& Role = “Sink”
		&& FAS_Pdu_Type (fas_pdu) <> “DTU_PDU”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAS",
		reason_code := " Invalid FAS-PDU",
		additional_detail := "null"
		}
			CLOSED

R4	OPEN	ErrorToARPM => (no actions taken)	OPEN

R5	OPEN	FAS-PDU_ind
		&& Role = “Source”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid Event for Role”,
		additional_detail := "null"
		}
			CLOSED

9.2.2 AR protocol machine (ARPM) for QUB AREP

9.2.2.1 QUB ARPM states

The defined states and their descriptions of the QUB ARPM are shown in Table 15 and Figure 25.

Table 15 – QUB ARPM states

CLOSED	The AREP is defined, but not capable of sending or receiving FAS-PDUs. It may send or receive Associate service FAS-PDUs while in this state.
OPEN	The AREP is defined and capable of sending or receiving FAS-PDUs.
REQUESTING (REQ)	The AREP has sent an Associate Request FAS-PDU and is waiting for a response from the remote AREP.
RESPONDING (RSP)	The AREP has received an Associate Request FAS-PDU, delivered an Associate.indication primitive and is waiting for a response from its user.

REPLIED (REPL)	The Server AREP has issued an ASC_rsp(+) primitive and is waiting for receiving a "connection-established" indication from the DLL.
SAME	Indicates that the next state is the same as the current state.

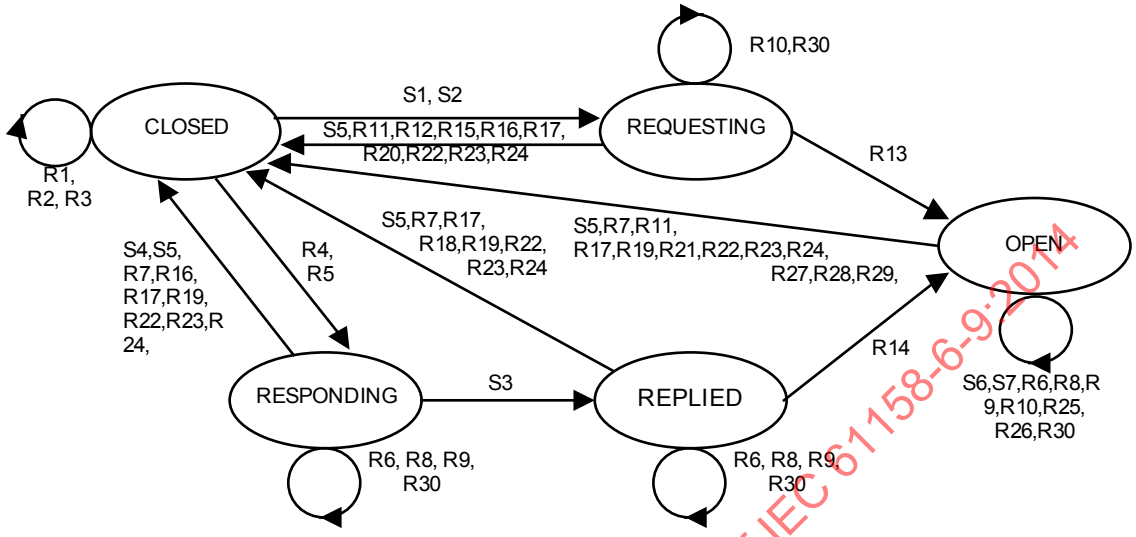


Figure 25 – State transition diagram of QUB ARPM

9.2.2.2 QUB ARPM state table

Table 16 and Table 17 specify the QUB ARPM state machine.

Table 16 – QUB ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1	CLOSED	ASC_req
		&& Initiator = "True"
		&& RemoteAddressConfigurationType := "Free"
		RemoteDlcepAddress := remote_dlcep_address,
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Connect_req",
		arep_id := GetArepId (),
		called_address := "default dlsap address",
		calling_address := "default dlsap address",
		local_dlcep_address := LocalDlcep,
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "ASC_ReqPDU",
		calling_dlcep_address := LocalDlcepAddress,
		called_dlcep_address := RemoteDlcepAddress,
		fas_data := user_data)
		}
			REQ

S2	CLOSED	ASC_req
		&& Initiator = "True"
		&& RemoteAddressConfigurationType := "Linked"
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Connect_req",
		arep_id := GetArepId (),
		called_address := "default dlsap address",
		calling_address := "default dlsap address",
		local_dlcep_address := LocalDlcep,
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "ASC_ReqPDU",
		calling_dlcep_address := LocalDlcepAddress,
		called_dlcep_address := RemoteDlcepAddress,
		fas_data := user_data)
		}
			REQ

#	Current state	Event or condition => action	Next state

S3	RSP	ASC_rsp(+)
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Connect_rsp",
		arep_id := GetArepId (),
		responding_address := “default dlsap address”,
		local_dlcep_address := LocalDlcep,
		dlcep_dl_id := DlcepDlIdentifier,
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "ASC_RspPDU",
		fas_data := user_data)
		}
			REPL

S4	RSP	ASC_rsp(-)
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "connection rejection–transient condition",
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "ASC_ErrPDU",
		fas_data := user_data)
		}
			CLOSED

S5	NOT CLOSED	Abort_req
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "disconnection–normal condition",
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "Abort_PDU",
		fas_id := identifer,
		fas_reason_code := reason_code,
		fas_additional_detail := additional_detail)
		}
			CLOSED

S6	OPEN	DTC_req
		&& Role = “Client” || “Peer”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "DTC_ReqPDU",
		fas_data := user_data)
		}
			OPEN

S7	OPEN	DTC_rsp
		&& Role = “Server” || “Peer”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "DTC_RspPDU",
		fas_data := user_data)
		}
			OPEN

Table 17 – QUB ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	CLOSED	Connect_ind
		&& Initiator = “True”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Multiple Initiators",
		dlsdu := “null”
		}
			CLOSED

R2	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAS_Pdu_Type (dls_user_data) <> “ASC_ReqPDU”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := “Invalid FAS-PDU”,
		dlsdu := “null”
		}
			CLOSED

R3	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAS_Pdu_Type (dls_user_data) = “ASC_ReqPDU”
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Remote Address Mismatch",
		dlsdu := “null”
		}
			CLOSED

R4	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAS_Pdu_Type (dls_user_data) = “ASC_ReqPDU”
		&& RemoteAddressConfigurationType = “Free”
		=>
		RemoteDlcepAddress := calling_dlcep_address,
		DlcepDlIdentifier := dlcep_dl_id,
		MaxDlsduSizeFromRequesterNegotiated := dlsdu_size_from_requester,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		
		ASC_ind {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			RSP

R5	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAS_Pdu_Type (dls_user_data) = “ASC_ReqPDU”
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlcepAddress = calling_dlcep_address
		=>
		DlcepDlIdentifier := dlcep_dl_id,
		MaxDlsduSizeFromRequesterNegotiated := dlsdu_size_from_requester,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		
		ASC_ind {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			RSP

#	Current state	Event or condition => action	Next state

R6	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Remote Address Mismatch",
		dlsdu := “null”
		}
			SAME

R7	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& RemoteDlcepAddress = calling_dlcep_address
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Invalid FAS-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAS",
		reason_code := "Invalid FAS-PDU"
		}
			CLOSED

R8	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& FAS_Pdu_Type (dls_user_data) = “ASC_ReqPDU”
		&& RemoteAddressConfigurationType = “Free”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "AREP Busy",
		dlsdu := “null”
		}
			SAME

R9	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& FAS_Pdu_Type (dls_user_data) <> “ASC_ReqPDU”
		&& RemoteAddressConfigurationType = “Free”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Invalid FAS-PDU",
		dlsdu := “null”
		}
			SAME

R10	REQ
	OPEN
		Connect_ind
		&& Initiator = “True”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Multiple Initiators",
		dlsdu := “null”
		}
			SAME

#	Current state	Event or condition => action	Next state

R11	REQ
	OPEN
		Connect_ind
		&& Initiator = “True”
		&& RemoteDlcepAddress = calling_dlcep_address
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Multiple Initiators",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAS",
		reason_code := "Multipe Initiators"
		}
			CLOSED

R12	REQ	Connect_cnf
		&& FAS_Pdu_Type (dls_user_data) <> “ASC_RspPDU”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid FAS-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAS",
		reason_code := "Invalid FAS-PDU"
		}
			CLOSED

R13	REQ	Connect_cnf
		&& FAS_Pdu_Type (dls_user_data) = “ASC_RspPDU”
		=>
		MaxDlsduSizeFromRequesterNegotiated := dlsdu_size_from_requester,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		DllPriorityNegotiated := dll_priority,
		
		ASC_cnf(+) {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			OPEN

R14	REPL	FAS-PDU_ind && dmpm_service_name = “DMPM_Connection_Established_ind” => (no actions taken)	OPEN

R15	REQ	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& FAS_Pdu_Type (fas_pdu) = “ASC_ErrPDU”
		=>
		ASC_cnf(-) {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			CLOSED

#	Current state	**Event** or **condition** **=> action**	Next state

R16	REQ
	RSP
		FAS-PDU_ind
		&& dmpm_service_name <> “DMPM_Disconnect_ind”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid DL-Event",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAS",
		reason_code := "Invalid DL-Event"
		}
			CLOSED

R17	NOT CLOSED	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fas_pdu <> “null”
		&& FAS_Pdu_Type (fas_pdu) = “Abort_PDU”
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "False",
		identifier := AbortIdentifier (fas_pdu),
		reason_code := AbortReason (fas_pdu),
		additional_detail := AbortDetail (fas_pdu)
		}
			CLOSED

R18	REPL	FAS-PDU_ind
		&& ((dmpm_service_name <> “DMPM_Disconnect_ind”)
		&& (dmpm_service_name <>“DM_Connection_Established_ind”))
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid DL-Event",
		dlsdu := “null”
		},
		
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid DL-Event”,
		additional_detail := “null”
		}
			CLOSED

R19	REPL
	RSP
	OPEN
		FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fas_pdu <> “null”
		&& FAS_Pdu_Type (fas_pdu) <> “Abort_PDU”
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid FAS-PDU”,
		additional_detail := “null”
		}
			CLOSED

#	Current state	Event or condition => action	Next state

R20	REQ	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fas_pdu <> “null”
		&& ((FAS_Pdu_Type (fas_pdu) <> “Abort_PDU”)
		&& (FAS_Pdu_Type (fas_pdu) <> “ASC_ErrPDU”))
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid FAS-PDU”,
		additional_detail := “null”
		}
			CLOSED

R21	OPEN	FAS-PDU_ind
		&& ((dmpm_service_name <> “DMPM_Disconnect_ind”)
		&& (dmpm_service_name <> “DMPM_Data_ind”))
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid DL-Event",
		dlsdu := “null”
		},
		
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid DL-Event”,
		additional_detail := “null”
		}
			CLOSED

R22	NOT CLOSED	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fas_pdu = "null"
		&& originator = "remote_dls_provider"
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "False",
		identifier := “Data-link layer”,
		reason_code := reason,
		additional_detail := "null"
		}
			CLOSED

R23	NOT CLOSED	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fas_pdu = "null"
		&& originator = "remote_dls_user"
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "False",
		identifier := “FAS”,
		reason_code := reason,
		additional_detail := "null"
		}
			CLOSED

R24	NOT CLOSED	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fas_pdu = "null"
		&& originator = "local_dls_provider"
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “Data-link layer”,
		reason_code := reason,
		additional_detail := "null"
		}
			CLOSED

#	Current state	**Event** or **condition** **=> action**	Next state

R25	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Peer” || “Server”
		&& FAS_Pdu_Type (fas_pdu) = “DTC_ReqPDU”
		=>
		DTC_ind {
		arep_id := GetArepId (),
		user_data := fas_pdu
		}
			OPEN

R26	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Client” || “Peer”
		&& FAS_Pdu_Type (fas_pdu) = “DTC_RspPDU”
		=>
		DTC_cnf {
		arep_id := GetArepId (),
		user_data := fas_pdu
		}
			OPEN

R27	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Server”
		&& FAS_Pdu_Type (fas_pdu) <> “DTC_ReqPDU”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid FAS-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid FAS-PDU”,
		additional_detail := "null"
		}
			CLOSED

R28	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Client”
		&& FAS_Pdu_Type (fas_pdu) <> “DTC_RspPDU”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid FAS-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid FAS-PDU”,
		additional_detail := "null"
		}
			CLOSED

#	Current state	Event or condition => action	Next state

R29	OPEN	FAS-PDU_ind
		&& Role = “Peer”
		&& dmpm_service_name = “DMPM_Data_ind”
		&& ((FAS_Pdu_Type (fas_pdu) <> “DTC_ReqPDU”)
		&& (FAS_Pdu_Type (fas_pdu) <> “DTC_RspPDU”))
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid FAS-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid FAS-PDU”,
		additional_detail := "null"
		}
			CLOSED

R30	NOT CLOSED	ErrorToARPM => (no actions taken)	SAME
NOTE It is a local matter to report an error status to network management entities. The ARPM does not abort the existing connections. The FAS user may issue an Abort request to disconnect the current connection, depending on the type of status information conveyed by the ErrorToARPM primitive.			

9.2.3 AR protocol machine (ARPM) for BNU AREP

9.2.3.1 BNU ARPM states

The defined states and their descriptions of the BNU ARPM are shown in Table 18 and Figure 26.

Table 18 – BNU ARPM states

CLOSED	The AREP is defined, but not capable of sending or receiving FAS-PDUs.
REQUESTING	The AREP has issued an ASC_req and waiting for an ASC_cnf primitive.
OPEN	The AREP is defined and capable of sending or receiving FAS-PDUs.

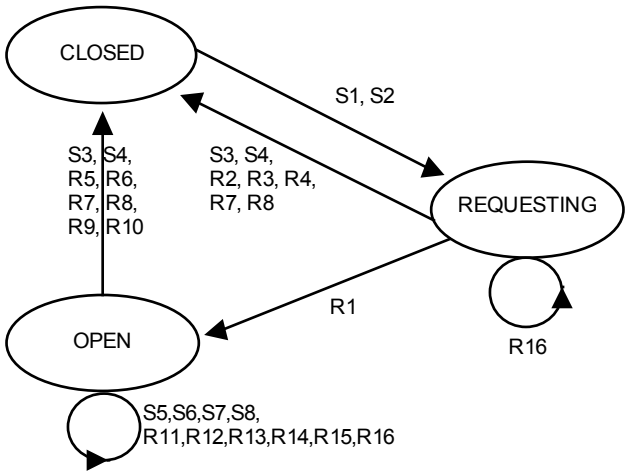


Figure 26 – State transition diagram of the BNU ARPM

9.2.3.2 BNU ARPM state table

Table 19 and Table 20 specify the BNU ARPM state machine.

Table 19 – BNU ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1	CLOSED	ASC_req
		&& Role = “Publisher”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Connect_req",
		arep_id := GetArepId (),
		called_address := “null”,
		calling_address := “default dlsap address”,
		local_dlcep_address := PublisherDlcepAddress,
		dlsdu := "null"
		}
			REQ

S2	CLOSED	ASC_req
		&& Role = “Subscriber”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Connect_req",
		arep_id := GetArepId (),
		called_address := PublisherDlcepAddress,
		calling_address := “default dlsap address”,
		local_dlcep_address := “default subscriber dlcep address”,
		dlsdu := "null"
		}
			REQ

S3	NOT CLOSED	Abort_req
		&& Role = “Publisher”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "disconnection–normal condition",
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := “Abort_PDU”,
		fas_id := identifer,
		fas_reason_code := reason_code,
		fas_additional_detail := additional_detail)
		}
			CLOSED

S4	NOT CLOSED	Abort_req
		&& Role = “Subscriber”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "disconnection–normal condition",
		dlsdu := “null”
		}
			CLOSED

S5	OPEN	DTU_req
		&& Role = “Publisher”
		=>
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Put_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAS-PDU (
		fas_pdu_name := "DTU_PDU",
		fas_sdu := user_data)
		}
			OPEN

S6	OPEN	FCMP_req
		&& Role = “Publisher”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Compel_Service_req”,
		arep_id := GetArepId (),
		action_class := “Local”
		}
			OPEN

#	Current state	Event or condition => action	Next state

S7	OPEN	FCMP_req
		&& Role = “Subscriber”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Compel_Service_req”,
		arep_id := GetArepId (),
		action_class := “Remote”
		}
			OPEN

S8	OPEN	GBM_req
		&& Role = “Subscriber”
		=>
		FAS-PDU_req {
		dmpm_service_name := “DMPM_Get_req”,
		arep_id := GetArepId ()
		}
			OPEN

Table 20 – BNU ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	REQ	Connect_cnf
		=>
		ASC_cnf(+) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			OPEN

R2	REQ	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& fas_pdu = “null”
		&& originator = "local_dls_provider"
		=>
		ASC_cnf(-) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			CLOSED

R3	REQ	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& fas_pdu = "null"
		&& originator = "remote_dls_provider"
		=>
		ASC_cnf(-) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			CLOSED

R4	REQ	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name <> "DMPM_Disconnect_ind"
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid DI Event”,
		additional_detail := "null"
		}
			CLOSED

R5	OPEN	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& fas_pdu = “null”
		&& originator = "local_dls_provider"
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “Data-link layer”,
		reason_code := reason,
		additional_detail := "null"
		}
			CLOSED

#	Current state	**Event** or **condition** **=> action**	Next state

R6	OPEN	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& dlsdu = "null"
		&& originator = "remote_dls_provider"
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "False",
		identifier := “Data-link layer,”
		reason_code := reason,
		additional_detail := “null”
		}
			CLOSED

R7	NOT CLOSED	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& Fas_Pdu_Type (fas_pdu) = “Abort_PDU”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "False",
		identifier := AbortIdentifer (fas_pdu),
		reason_code := AbortReason (fas_pdu),
		additional_detail := AbortDetail (fas_pdu)
		}
			CLOSED

R8	NOT CLOSED	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& fas_pdu <> “null”
		&& Fas_Pdu_Type (fas_pdu) <> “Abort_PDU”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAS”,
		reason_code := “Invalid FAS-PDU”,
		additional_detail := "null"
		}
			CLOSED

R9	OPEN	FAS-PDU_ind
		&& Role = “Subscriber”
		&& ((dmpm_service_name <> "DMPM_Buffer_Received_ind")
		&& (dmpm_service_name <> "DMPM_Disconnect_ind”)
		&& (dmpm_service_name <> "DMPM_Compel_Service_cnf”)
		&& (dmpm_service_name <> "DMPM_Get_cnf”)
		&& (dmpm_service_name <> "DMPM_Buffer_Sent_ind”))
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAS",
		reason_code := "Invalid DI Event",
		additional_detail := "null"
		},
		
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid DL-Event",
		dlsdu := "null"
		}
			CLOSED

#	Current state	Event or condition => action	Next state

R10	OPEN	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Buffer_Received_ind"
		&& FAS_pdu_type <> "DTU_PDU"
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAS",
		reason_code := "Invalid FAS-PDU",
		additional_detail := "null"
		},
		
		FAS-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		dlcep_dl_id := DlcepDlIdentifier,
		reason := "Invalid FAS-PDU",
		dlsdu := "null"
		}
			CLOSED

R11	OPEN	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = "DMPM_Buffer_Received_ind"
		&& FAS_Pdu_Type (fas_pdu) = "DTU_PDU"
		=>
		DTU_ind {
		arep_id := GetArepId (),
		duplicate_fas_sdu := duplicate_dlsdu,
		user_data := fas_pdu,
		local_timeliness := local_dle_timeliness,
		remote_timeliness := remote_dle_timeliness
		}
			OPEN

R12	OPEN	FAS-PDU_ind
		&& dmpm_service_name = “DMPM_Compel_Service_cnf”
		=>
		FCPM_cnf {
		status := dl_status
		}
			OPEN

R13	OPEN	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = “DMPM_Get_cnf”
		&& status = “success”
		=>
		GBM_cnf(+) {
		arep_id := GetArepId (),
		duplicate_fas_sdu := duplicate_dlsdu,
		user_data := fas_pdu,
		local_timeliness := local_dle_timeliness,
		remote_timeliness := remote_dle_timeliness
		}
			OPEN

R14	OPEN	FAS-PDU_ind
		&& Role = “Subscriber”
		&& dmpm_service_name = “DMPM_Get_cnf”
		&& status <> “success”
		=>
		GBM_cnf(-) {
		arep_id := GetArepId ()
		}
			OPEN

R15	OPEN	FAS-PDU_ind
		&& Role = “Publisher”
		&& dmpm_service_name = “DMPM_Buffer_Sent_ind”
		=>
		FSTS_ind {
		arep_id := GetArepId (),
		reported_status := “Buffer-Sent”
		}
			OPEN

R16	NOT CLOSED	ErrorToARPM => (no actions taken)	SAME
NOTE It is a local matter to report an error status to network management entities. The ARPM does not abort the existing connections. The FAS user may issue an Abort request to disconnect the current connection, depending on the type of status information conveyed by the ErrorToARPM primitive.			

9.3 AREP state machine primitive definitions

9.3.1 Primitives exchanged between DMPM and ARPM

The primitives exchanged between the DMPM and the ARPM are specified in Table 21 and Table 22.

Table 21 – Primitives issued from ARPM to DMPM

Primitive names	Source	Associated parameters	Functions
FAS-PDU_req	ARPM	dmpm_service_name, arep_id, dlcep_dl_id, called_address, calling_address, responding_address, local_dlcep_addres, reason, dlsdu	This primitive is used to request the DMPM to transfer an FAS-PDU, or to request an abort without transferring an FAS-PDU. It passes the FAS-PDU to the DMPM as a DLSDU. It also carries some of the Data Like Layer parameters that are referenced there.

Table 22 – Primitives issued by DMPM to ARPM

Primitive names	Source	Associated parameters	Functions
FAS-PDU_ind	DMPM	dmpm_service_name, originator, reason, duplicate_dlsdu, calling_address, dll_priority, fas_pdu, local_dle_timeliness, remote_dle_timeliness	This primitive is used to pass an FAS-PDU received as a Data Like Layer service data unit to a designated ARPM. It also carries some of the Data Like Layer parameters that are referenced in the ARPM.
ErrorToARPM	DMPM	originator, reason	This primitive is used to convey selected communication errors reported by the Data Like Layer to a designated ARPM.
Connect_ind	DMPM	dlcep_dl_id, calling_dlcep_address, called_dlcep_address, called_address, calling_address, dlcep_class, delivery_from_responder, delivery_from_requester, dll_priority, max_confirm_delay_on_connect, max_confirm_delay_on_data, dlsdu_size_from_requester, dlsdu_size_from_responder, dls_user_data	This primitive is used to convey a DL_Connect.ind primitive to the ARPM to process connection establishment. All the parameters that are associated with the DL_Connect.ind primitive are carried with this primitive.
Connect_cnf	DMPM	responding_address, dlcep_class, delivery_from_responder, delivery_from_requester, dll_priority, max_confirm_delay_on_connect, max_confirm_delay_on_data, dlsdu_size_from_requester, dlsdu_size_from_responder, dls_user_data	This primitive is used to convey a DL_Connect.cnf primitive to the ARPM. All the parameters that are associated with the DL_Connect.cnf are carried with this primitive.

9.3.2 Parameters of ARPM/DMPM primitives

The parameters used with the primitives exchanged between the ARPM and the DMPM are described in Table 23.

Table 23 – Parameters used with primitives exchanged between ARPM and DMPM

Parameter name	Description
dmpm_service_name	This parameter conveys a Data Like Layer primitive name. Possible values are all the DL-XXXX.yyy primitives defined in this section and are represented as DMPM_XXXX_yyy.
originator	This parameter conveys the value of the dl_originator parameter of the received primitive.

Parameter name	Description
reason	This parameter conveys the value of the dl_reason parameter.
duplicate_dlsdu	This parameter conveys the value of the dl_duplicate_dlsdu parameter of the received primitive.
calling_address	This parameter conveys the value of the dl_calling_address parameter of the received primitive.
dll_priority	This parameter conveys the value of the dl_dll_priority parameter of the received primitive.
fas_pdu	This parameter conveys the value of the dl_dls_user_data parameter of the received primitive.
dlcep_dl_id	This parameter conveys the value of the dl_dlcep_dl_id parameter supplied with the Data Like Layer primitive.
calling_dlcep_address	This parameter conveys the value of the RequestingAREP parameter supplied with the ASC_ReqPDU.
called_dlcep_address	This parameter conveys the value of the RespondingAREP parameter supplied with the ASC_ReqPDU.
called_address	This parameter conveys the value of the dl_called_address parameter of the received primitive.
dlcep_class	This parameter conveys the value of the dl_dlcep_class parameter of the received primitive.
delivery_from_responder	This parameter conveys the value of the dl_delivery_from_responder parameter of the received primitive.
delivery_from_requester	This parameter conveys the value of the dl_delivery_from_requester parameter of the received primitive.
max_confirm_delay_on_ connect	This parameter conveys the value of the dl_max_confirm_delay_on_connect parameter of the received primitive.
max_confirm_delay_on_data	This parameter conveys the value of the dl_max_confirm_delay_on_data parameter of the received primitive.
dlsdu_size_from_requester	This parameter conveys the value of the dl_dlsdu_size_from_requester parameter of the received primitive.
dlsdu_size_from_responder	This parameter conveys the value of the dl_dlsdu_size_from_responder parameter of the received primitive.
dls_user_data	This parameter conveys the value of the dl_dls_user_data parameter of the received primitive.
responding_address	This parameter conveys the value of the dl_responding_address parameter of the received primitive.

9.4 AREP state machine functions

Table 24 through Table 29 define the functions used by the ARPM.

Table 24 – Function GetArepId()

Name	GetArepId ()	Used in	ARPM
Input		Output	
(none)		AREP Identifier	
Function			
Returns a value that can unanimously identify the current AREP.			

Table 25 – Function BuildFAS-PDU

Name	BuildFAS-PDU	Used in	ARPM
Input		Output	
fas_pdu_name, calling_dlcep_address, called_dlcep_address, fas_data, fas_id, fas_reason_code, fas_additional_detail		dlsdu	
Function			
Builds an FAS-PDU out of the parameters given as input variables.			

Table 26 – Function FAS_Pdu_Type

Name	FAS_Pdu_Type	Used in	ARPM
Input		Output	
dls_user_data		One of the FAS-PDU types defined in the FAS-PDUs section.	
Function			
This function decodes the FAS-PDU that is conveyed in the dls_user_data parameter and retrieves one of the FAS-PDU types.			

Table 27 – Function AbortIdentifier

Name	AbortIdentifier	Used in	ARPM
Input		Output	
fas_pdu		The Identifier parameter of the Abort service.	
Function			
This function decodes the Abort_PDU that is conveyed in the fas_pdu parameter and extracts the Identifier parameter.			

Table 28 – Function AbortReason

Name	AbortReason	Used in	ARPM
Input		Output	
fas_pdu		The Reason Code parameter of the Abort service.	
Function			
This function decodes the Abort_PDU that is conveyed in the fas_pdu parameter and extracts the Reason Code parameter.			

Table 29 – Function AbortDetail

Name	AbortDetail	Used in	ARPM
Input		Output	
fas_pdu		The Additional Detail parameter of the Abort service.	
Function			
This function decodes the Abort_PDU that is conveyed in the fas_pdu parameter and extracts the Additional Detail parameter.			

10 DLL mapping protocol machine (DMPM)

10.1 DMPM States

The defined states and their descriptions of the DMPM are shown in Table 30 and Figure 27.

Table 30 – DMPM state descriptions

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

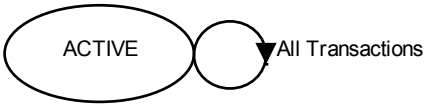


Figure 27 – State transition diagram of DMPM

10.2 DMPM state table

Table 31 and Table 32 specify the DMPM state machine.

Table 31 – DMPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = "DMPM_Connect_req"
		=>
		PickArep (arep_id),
		
		DL_Connect.req {
		calling_address := calling_address,
		dl_dlcep_address := local_dlcep_address,
		dl_dlcep_class := DlcepClass,
		dl_delivery_from_requester := FromRequesterToResponder, *1
		dl_delivery_from_responder := FromResponderToRequester, *2
		dl_dll_priority := DllPriority,
		dl_max_confirm_delay_on_connect := MaxConfirmDelayOnDlConnect,
		dl_max_confirm_delay_on_data := MaxConfirmDelayOnDlData, *1, *2
		dl_dlpdu_authentication := DlpduAuthentication,
		dl_residual_activity_as_sender := ResidualActivityAsSender, *1
		dl_residual_activity_as_receiver := ResidualActivityAsReceiver, *1, *2
		dl_scheduling_policy := DlSchedulingPolicy,
		dl_dlsdu_size_from_requester := MaxDlsduSizeFromRequester, *1
		dl_dlsdu_size_from_responder := MaxDlsduSizeFromResponder, *2
		dl_buffer_queue_binding_as_sender := SendingBufferOrQueueIdentifier, *1
		dl_buffer_queue_binding_as_receiver := ReceivingBufferOrQueueIdentifier, *2
		dl_dls_user_data := dlsdu, *1, *2
		dl_sender_dl_timeliness_class := SenderDlTimelinessClass, *4
		dl_sender_time_window_size := SenderTimeWindowSize, *4
		dl_sender_synchronizing_dlcep := SenderSynchronizingDlcep, *4
		dl_receiver_dl_timeliness_class := ReceiverDlTimelinessClass, *3
		dl_receiver_time_window_size := ReceiverTimeWindowSize, *3
		dl_receiver_synchronizing_dlcep := ReceiverSynchronizingDlcep *3
		DlcepDlIdentifier := dl_dlcep_dl_id (from DL_Connect.req(out))
		}
		
		*1: Not used with BNU AREPs whose role is Subscriber.
		*2: Not used with BNU AREPs whose role is Publisher.
		*3: Only used with BNU AREPs whose role is Subscriber.
		*4: Only used with BNU AREPs whose role is Publisher.
			ACTIVE

#	Current state	Event or condition => action	Next state

S2	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = "DMPM_Connect_rsp"
		=>
		PickArep (arep_id),
		
		DL_Connect.rsp {
		dl_dlcep_dl_id := dlcep_dl_id,
		dl_responding_address := responding_address,
		dl_dlcep_address := local_dlcep_address,
		dl_dlcep_class := DlcepClass,
		dl_delivery_from_requester := FromRequesterToResponder,
		dl_delivery_from_responder := FromResponderToRequester,
		dl_dll_priority := DllPriority,
		dl_max_confirm_delay_on_connect := MaxConfirmDelayOnDlConnect,
		dl_max_confirm_delay_on_data := MaxConfirmDelayOnDlData,
		dl_dlpdu_authentication := DlpduAuthentication,
		dl_residual_activity_as_sender := ResidualActivityAsSender,
		dl_residual_activity_as_receiver := ResidualActivityAsReceiver,
		dl_scheduling_policy := DlSchedulingPolicy,
		dl_dlsdu_size_from_requester := MaxDlsduSizeFromRequesterNegotiated,
		dl_dlsdu_size_from_responder := MaxDlsduSizeFromResponderNegotiated,
		dl_buffer_queue_binding_as_sender := SendingBufferOrQueueIdentifier,
		dl_buffer_queue_binding_as_receiver := ReceivingBufferOrQueueIdentifier,
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S3	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = "DMPM_Disconnect_req"
		=>
		PickArep (arep_id),
		
		DL_Disconnect.req {
		dl_dlcep_dl_id := dlcep_dl_id,
		dl_reason := reason,
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S4	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = "DMPM_Data_req"
		=>
		PickArep (arep_id),
		
		dmpm_request_id := SelectIdentifier(),
		
		DL_Data.req {
		dl_request_dls_user_id := dmpm_request_id,
		dl_dlcep_dl_id := dlcep_dl_id,
		dl_dls_user_data := dlsdu,
		}
			ACTIVE

S5	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = "DMPM_Unitdata_req"
		=>
		dmpm_request_id := SelectIdentifier(),
		
		PickArep (arep_id),
		
		DL_Unitdata.req {
		dl_request_dls_user_id := dmpm_request_id,
		dl_called_address := RemoteDlsapAddress,
		dl_calling_address := LocalDlsapAddress,
		dl_dll_priority := DllPriority,
		dl_max_confirm_delay := MaxConfirmDelayOnUnitdata,
		dl_remote_dle_confirmed := "False",
		dl_dls_user_data := dlsdu
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

S6	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = "DMPM_Put_req"
		=>
		PickArep (arep_id),
		
		DL_Put.in {
		dl_buffer_id := SendingBufferOrQueueIdentifier,
		dl_dls_user_data := dlsdu
		}
		
		DL_Put.out
		&& dl_status = "success"
		
		(no actions taken)
		
		DL_Put.out
		&& dl_status <> "success"
		
		ErrorToARPM {
		originator := "local_dls",
		reason := dl_status
		}
			ACTIVE

S7	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = “DMPM_Get_req”
		=>
		PickArep (arep_id),
		
		DL_Get.in {
		dl_buffer_or_queue_id := ReceivingBufferOrQueueIdentifier
		}
		
		DL_Get.out
		&& dl_reported_service_identification_class = "NONE"
		
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Get_cnf",
		status := dl_status,
		duplicate_dlsdu := dmpm_duplicate_dlsdu,
		fas_pdu := dl_dls_user_data,
		local_dle_timeliness := dl_local_dle_timeliness,
		remote_dle_timeliness := dl_sender_and_remote_dle_timeliness
		}
			ACTIVE

S8	ACTIVE	FAS-PDU_req
		&& dmpm_service_name = “DMPM_Compel_req”
		=>
		PickArep (arep_id),
		
		DL_Compel_Service.in {
		dl_action_class := action_class,
		dl_dlcep_dl_id := DlcepDlIdentifier
		}
		
		DL_Compel_Service.out
		
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Compel_Service_cnf",
		status := dl_status
		}
			ACTIVE

Table 32 – DMPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	ACTIVE	DL_Connect.ind
		&& LocateQubArep (dl_dls_user_data) = “NOT FOUND”
		=>
		DL_Disconnect.req {
		dl_dlcep_dl_id := dl_dlcep_dl_id (from indication primitive),
		dl_reason := “AREP Not Found”,
		dl_dls_user_data := “null”
		}
			ACTIVE

R2	ACTIVE	DL_Connect.ind
		&& LocateQubArep (dl_dls_user_data) = “NOT QUB”
		=>
		DL_Disconnect.req {
		dl_dlcep_dl_id := dl_dlcep_dl_id (from indication primitive),
		dl_reason := “Invalid AREP type”,
		dl_dls_user_data := “null”
		}
			ACTIVE

R3	ACTIVE	DL_Connect.ind
		&& LocateQubArep (dl_dls_user_data) = “True”
		=>
		Connect_ind {
		dlcep_dl_id := dl_dlcep_dl_id,
		calling_dlcep_address := RequestingAREP,
		called_dlcep_address := RespondingAREP,
		called_address := dl_called_address,
		calling_address := dl_calling_address,
		dlcep_class := dl_dlcep_class,
		delivery_from_requester := dl_delivery_from_requester,
		delivery_from_responder := dl_delivery_from_responder,
		dll_priority := dl_dll_priority,
		max_confirm_delay_on_connect := dl_max_confirm_delay_on_connect,
		max_confirm_delay_on_data := dl_max_confirm_delay_on_data,
		dlsdu_size_from_requester := dl_dlsdu_size_from_requester,
		dlsdu_size_from_responder := dl_dlsdu_size_from_responder,
		dls_user_data := dl_dls_user_data
		}
			ACTIVE

R4	ACTIVE	DL_Connect.cnf
		&& FindAREP (dl_dlcep_dl_id) = "False"
		=>
		DL_Disconnect.req {
		dl_dlcep_dl_id := dl_dlcep_dl_id (from confirmation primitive),
		dl_reason := “DLCEP Not Found”,
		dl_dls_user_data := “null”
		}
			ACTIVE

R5	ACTIVE	DL_Connect.cnf
		&& FindAREP (dl_dlcep_dl_id) = "True"
		=>
		Connect_cnf {
		responding_address := dl_responding_address,
		dlcep_class := dl_dlcep_class,
		delivery_from_requester := dl_delivery_from_requester, *1
		delivery_from_responder := dl_delivery_from_responder, *2
		dll_priority := dl_dll_priority,
		max_confirm_delay_on_connect := dl_max_confirm_delay_on_connect,
		max_confirm_delay_on_data := dl_max_confirm_delay_on_data,
		dlsdu_size_from_requester := dl_dlsdu_size_from_requester, *1
		dlsdu_size_from_responder := dl_dlsdu_size_from_responder, *2
		dls_user_data := dl_dls_user_data *1
		}
		
		*1: Not used with BNU AREPs whose role is Subscriber.
		*2: Not used with BNU AREPs whose role is Publisher.
			ACTIVE

R6	ACTIVE	DL_Connection_Established.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state

R7	ACTIVE	DL_Connection_Established.ind
		&& FindAREP (dl_dlcep_dl_id) = "True"
		=>
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Connection_Established_ind",
		fas_pdu := "null"
		}
			ACTIVE

R8	ACTIVE	DL_Disconnect.ind && FindAREP (dl_dlcep_dl_id) = “False” => (no actions taken)	ACTIVE

R9	ACTIVE	DL_Disconnect.ind
		&& FindAREP (dl_dlcep_dl_id) = "True"
		=>
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Disconnect_ind",
		originator := dl_originator,
		reason := dl_reason,
		fas_pdu := dl_dls_user_data
		}
			ACTIVE

R10	ACTIVE	DL_Unitdata.ind && FindAREP (dl_called_address) = "False" => (no actions taken)	ACTIVE

R11	ACTIVE	DL_Unitdata.ind
		&& FindAREP (dl_called_address) = "True"
		&& ExplicitQueue = "False"
		=>
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Unitdata_ind",
		calling_address := dl_calling_address,
		dll_priority := dl_dll_priority,
		fas_pdu := dl_dls_user_data
		}
			ACTIVE

R12	ACTIVE	DL_Unitdata.ind
		&& FindAREP (dl_called_address) = "True"
		&& ExplicitQueue = "True"
		=>
		DL_Get.in {
		dl_buffer_or_queue_dl_id := QueueIdentifier
		}
		
		DL_Get.out
		&& dl_status = "success"
		&& dl_reported_service_identification_class = "DL(SAP)-ADDRESS"
		&& dl_called_dl(sap)_addresss = LocalDlsapAddress
		
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Unitdata_ind",
		calling_address := dl_calling_address,
		dll_priority := dl_dll_priority,
		fas_pdu := dl_dls_user_data
		}
		
		DL_Get.out
		&& dl_status <> "success"
		
		ErrorToARPM {
		originator := "local_dls",
		reason := dl_status
		}
			ACTIVE

R13	ACTIVE	DL_Unitdata.cnf
		&& dmpm_request_id = dl_request_dls_user_id
		&& dl_status <> "success"
		=>
		ErrorToARPM {
		originator := "local_dls",
		reason := dl_status
		}
			ACTIVE

R14	ACTIVE	DL_Unitdata.cnf && dmpm_request_id = dl_request_dls_user_id && dl_status = "success" => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state
R15	ACTIVE	DL_Data.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE

R16	ACTIVE	DL_Data.ind
		&& FindAREP (dl_dlcep_dl_id) = "True"
		&& ExplicitQueue = "False"
		=>
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Data_ind",
		fas_pdu := dl_dls_user_data
		}
			ACTIVE

R17	ACTIVE	DL_Data.ind
		&& FindAREP (dl_dlcep_address) = "True"
		&& ExplicitQueue = "True"
		=>
		DL_Get.in {
		dl_buffer_or_queue_id := ReceivingBufferOrQueueIdentifier
		}
		
		DL_Get.out
		&& dl_status = "success"
		&& dl_reported_service_identification_class = "DLCEP"
		
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Data_ind",
		fas_pdu := dl_dls_user_data
		}
		
		DL_Get.out
		&& dl_status <> "success"
		
		ErrorToARPM {
		originator := "local_dls",
		reason := dl_status
		}
			ACTIVE

R18	ACTIVE	DL_Data.cnf && dmpm_request_id <> dl_request_dls_user_id => (no actions taken)	ACTIVE
R19	ACTIVE	DL_Data.cnf && dmpm_request_id = dl_request_dls_user_id && dl_status = "success" => (no actions taken)	ACTIVE

R20	ACTIVE	DL_Data.cnf
		&& dmpm_request_id = dl_request_dls_user_id
		&& dl_status <> "success"
		=>
		ErrorToARPM {
		originator := "local_dls",
		reason := dl_status,
		dlsdu := "null"
		}
			ACTIVE

R21	ACTIVE	DL_Buffer_Received.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state

R22	ACTIVE	DL_Buffer_Received.ind
		&& FindAREP (dl_dlcep_dl_id) = "True"
		=>
		dmpm_duplicate_dlsdu := dl_duplicate_dlsdu,
		
		DL_Get.in {
		dl_buffer_or_queue_id := ReceivingBufferOrQueueIdentifier
		}
		
		DL_Get.out
		&& dl_status = "success"
		&& dl_reported_service_identification_class = "NONE"
		
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Buffer_Received_ind",
		duplicate_dlsdu := dmpm_duplicate_dlsdu,
		fas_pdu := dl_dls_user_data,
		local_dle_timeliness := dl_local_dle_timeliness,
		remote_dle_timeliness := dl_sender_and_remote_dle_timeliness
		}
		
		DL_Get.out
		&& dl_status <> "success"
		
		ErrorToARPM {
		originator := "local_dls",
		reason := dl_status
		}
			ACTIVE

R23	ACTIVE	DL_Buffer_Sent.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE

R24	ACTIVE	DL_Buffer_Sent.ind
		&& FindAREP (dl_dlcep_dl_id) = "True"
		=>
		FAS-PDU_ind {
		dmpm_service_name := "DMPM_Buffer_Sent_ind"
		}
			ACTIVE

10.3 Primitives exchanged between data-link layer and DMPM

The primitives exchanged between the data-link layer and the DMPM are specified in Table 33. They are defined in IEC 61158-3-1, and are prefixed by "dl_" to indicate that they are defined by the DLL.

Table 33 – Primitives exchanged between data-link layer and DMPM

Primitive names	Source	Associated parameters
DL_Connect.ind	Data-link layer	dl_dlcep_dl_id, dl_called_address, dl_calling_address, dl_dlcep_class, dl_delivery_from_requester, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlsdu_size_from_requester, dl_dlsdu_size_from_responder, dl_dls_user_data
DL_Connect.req(out)	Data-link layer	dl_dlcep_dl_id

Primitive names	Source	Associated parameters
DL_Connect.cnf		dl_dlcep_dl_id, dl_responding_address, dl_dlcep_class, dl_delivery_from_requester, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlsdu_size_from_requester, dl_dlsdu_size_from_responder, dl_sender_dl_timeliness_class, dl_receiver_dl_timeliness_class, dl_dls_user_data
DL_Connection_Established.ind	Data-link layer	dl_dlcep_dl_id,
DL_Disconnect.ind	Data-link layer	dl_dlcep_dl_id, dl_originator, dl_reason, dl_dls_user_data
DL_Data.ind	Data-link layer	dl_dlcep_dl_id, dl_dls_user_data
DL_Data.cnf	Data-link layer	dl_request_dls_user_id, dl_status
DL_Buffer_Received.ind	Data-link layer	dl_dlcep_dl_id, dl_duplicate_dlsdu
DL_Unitdata.ind	Data-link layer	dl_called_address, dl_calling _address, dl_dll_priority, dl_dls_user_data
DL_Unitdata.cnf	Data-link layer	dl_request_dls_user_id, dl_status
DL_Buffer_Sent.ind	Data-link layer	dl_dlcep_dl_id
DL_Get.out	Data-link layer	dl_status, dl_reported_service_identification_class, dl_receiving_dlcep_address, dl_called_dl(sap)_address, dl_calling_dlsap_address, dl_dll_priority, dl_dls_user_data, dl_local_dle_timeliness, dl_sender_and_remote_dle_timeliness
DL_Put.out	Data-link layer	dl_status
DL_Compel_Service.out	Data-link layer	dl_status
DL_Connect.req(in)	DMPM	dl_called_address, dl_calling_address, dl_dlcep_address, dl_dlcep_class, dl_delivery_from_requester, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlpdu_authentication, dl_scheduling_policy, dl_dlsdu_size_from_requester, dl_dlsdu_size_from_responder, dl_buffer_queue_binding_as_sender, dl_buffer_queue_binding_as_receiver, dl_sender_dl_timeliness_class, dl_sender_time_window_size, dl_sender_synchronizing_dlcep, dl_receiver_dl_timeliness_class, dl_receiver_time_window_size, dl_receiver_synchronizing_dlcep, dl_dls_user_data

Primitive names	Source	Associated parameters
DL_Connect.rsp	DMPM	dl_dlcep_dl_id, dl_responding_address, dl_dlcep_address, dl_dlcep_class, dl_delivery_from_requester, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlpdu_authentication, dl_scheduling_policy, dl_dlsdu_size_from_requester, dl_dlsdu_size_from_responder, dl_buffer_queue_binding_as_sender, dl_buffer_queue_binding_as_receiver, dl_dls_user_data
DL_Unitdata.req	DMPM	dl_request_dls_user_id, dl_called_address, dl_calling _address, dl_dll_priority, dl_max_confirm_delay, dl_remote_dle_confirm, dl_dls_user_data
DL_Disconnect.req	DMPM	dl_dlcep_dl_id, dl_reason dl_dls_user_data
DL_Data.req	DMPM	dl_request_dls_user_id, dl_dlcep_dl_id, dl_dls_user_data
DL_Get.in	DMPM	dl_buffer_or_queue_id
DL_Put.in	DMPM	dl_buffer_id, dl_dls_user_data
DL_Compel_Service.in	DMPM	dl_action_class, dl_dlcep_dl_id

10.4 Functions used by DMPM

Table 34 through Table 37 define internal functions used by the three DMPMs.

Table 34 – Function PickArep

Name	PickArep	Used in	DMPM
Input		Output	
arep_id		(all the attributes of the specified AREP)	
Function			
Selects the attributes for the AREP specified by the arep_id parameter. After this function is executed, the attributes of the selected AREP are available to the state machine.			

Table 35 – Function FindAREP

Name	FindAREP	Used in	DMPM
Input		Output	
dl_called_address || dl_dlcep_dl_id		True || False	
Function			
Check if the AREP that is specified either by the dl_called_address or dl_dlcep_dl_id parameters exists or not. True means the AREP exists. If it does, this function also returns a means to send a DMPM primitive to that AREP.			

Table 36 – Function LocateQubArep

Name	LocateQubArep	Used in	DMPM
Input		Output	
dl_dls_user_data (of DL_Connect.ind)		True || Not Found || Not QUB Access to attributes of the located AREP.	
Function			
This function returns the value of True and a means to get access to attributes of the located AREP if all of the following conditions are met. Otherwise, it returns the value of either Not Found or Not QUB. a) Decodes the DLSDU that is conveyed by the dl_dls_user_data argument and checks if the FAS PDU type is "ASC_ReqPDU." b) Decodes the FAS-PDU and extracts the RequestingAREP and RespondingAREP parameters. c) Looks for the QUB AREP whose LocalDlcepAddress attribute value is equal to the RespondingAREP.			

Table 37 – Function SetIdentifier()

Name	SetIdentifier()	Used in	DMPM
Input		Output	
(none)		Data Like Layer identifier	
Function			
This function picks an identifier that is used to uniquely identify Data Like Layer primitives.			

Bibliography

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

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

ISO/IEC 2375, *Information technology – Procedure for registration of escape sequences and coded character sets*

SOMMAIRE

AVANT-PROPOS 99
INTRODUCTION 101
1 Domaine d'application 102
1.1 Généralités 102
1.2 Spécifications 102
1.3 Conformité 103
2 Références normatives 103
3 Termes, définitions, symboles, abréviations et conventions 104
3.1 Termes et définitions provenant d'autres normes ISO/CEI 104
3.2 Termes de la CEI 61158-1 105
3.3 Abréviations et symboles 109
3.4 Conventions 109
3.5 Conventions utilisées dans les diagrammes d'états 110
4 Syntaxe abstraite 111
4.1 Syntaxe abstraite des unités PDU FAL-AR 111
4.2 Syntaxe abstraite de PDUBody 113
4.3 Définitions de types d'éléments ASE 117
4.4 Syntaxe abstraite des data types 122
5 Syntaxe de transfert 123
6 Structure des diagrammes d'états de protocole de la couche FAL 133
7 Diagrammes d'états de contexte AP 135
7.1 Structure PM VCR 135
7.2 Diagramme d'états de la machine PM VCR 135
8 Machine de protocole de service FAL (FSPM) 148
8.1 Généralités 148
8.2 Tableaux d'états de la machine FSPM 148
8.3 Fonctions utilisées par la machine FSPM 151
8.4 Paramètres des primitives FSPM/ARPM 151
9 Machines de protocole de relations entre applications (ARPM) 151
9.1 Mapping de point AREP avec la couche Liaison de données 151
9.2 Machines de protocole de relations entre applications (ARPM) 162
9.3 Définition des primitives du diagramme d'états AREP 177
9.4 Fonctions du diagramme d'états AREP 178
10 Machine de protocole de mapping de couche DLL (DMPM) 180
10.1 Etats de la machine DMPM 180
10.2 Tableau d'états de la machine DMPM 180
10.3 Primitives échangées entre la couche Liaison de données et la machine DMPM 187
10.4 Fonctions utilisées par la machine DMPM 189
Bibliographie 191

Figure 1 – Insertion d'informations d'identification dans l'unité PDU FMS 123
Figure 2 – Identification 124
Figure 3 – Codage avec identification 125

Figure 4 – Codage sans identification 126
Figure 5 – Représentation de la valeur true 126
Figure 6 – Représentation de la valeur false 126
Figure 7 – Codage des données de type Integer16 126
Figure 8 – Codage des données de type Unsigned16 127
Figure 9 – Codage des données de type Floating Point 127
Figure 10 – Codage des données de type Visible String 127
Figure 11 – Codage des données de type Octet String 128
Figure 12 – Codage des données de type Date 128
Figure 13 – Codage des données de type Time-of-day 129
Figure 14 – Codage des données de type Time-difference 129
Figure 15 – Codage des données de type Bit String 130
Figure 16 – Codage des données de type Time-value 130
Figure 17 – Codage des données de définitions de données d'utilisateur avec identificateur 131
Figure 18 – Codage des données de définitions de données d'utilisateur sans identificateur 131
Figure 19 – Codage de l'ID Info d'une structure SEQUENCE 132
Figure 20 – Relations entre les machines de protocole et les couches adjacentes 134
Figure 21 – Relations entre les machines de protocole et les couches adjacentes 135
Figure 22 – Diagramme d'états VCR 136
Figure 23 – Schéma de transition d'états de FSPM 148
Figure 24 – Diagramme de passages d'état de la machine ARPM QUU 162
Figure 25 – Diagramme de passages d'état de la machine ARPM QUB 164
Figure 26 – Diagramme de passages d'état de la machine ARPM BNU 173
Figure 27 – Schéma de transition d'états de DMPM 180

Tableau 1 – Conventions utilisées dans les diagrammes d'états 110
Tableau 2 – Codage du type Date 128
Tableau 3 – Transactions du diagramme d'états VCR AP 137
Tableau 4 – Primitives adressées par l'utilisateur FAL à la machine PM VCR 146
Tableau 5 – Primitives adressées par la machine PM VCR à l'utilisateur FAL 146
Tableau 6 – Primitives adressées par la machine PM VCR à la machine FSPM 147
Tableau 7 – Primitives adressées par la machine FSPM à la machine PM VCR 147
Tableau 8 – Tableau d'états de la machine FSPM: transactions expéditeur 149
Tableau 9 – Tableau d'états de la machine FSPM: transactions destinataire 150
Tableau 10 – Fonction SelectArep() 151
Tableau 11 – Paramètres utilisés avec les primitives échangées entre la machine FSPM et la machine ARPM 151
Tableau 12 – Etats de la machine ARPM QUU 162
Tableau 13 – Tableau d'états de la machine ARPM QUU: transactions expéditeur 162
Tableau 14 – Tableau d'états de la machine ARPM QUU: transactions destinataire 163
Tableau 15 – Etats de la machine ARPM QUB 164
Tableau 16 – Tableau d'états de la machine ARPM QUB: transactions expéditeur 165

Tableau 17 – Tableau d'états de la machine ARPM QUB: transactions destinataire 166
Tableau 18 – Etats de la machine ARPM BNU 172
Tableau 19 – Tableau d'états de la machine ARPM BNU: transactions expéditeur 173
Tableau 20 – Tableau d'états de la machine ARPM BNU: transactions destinataire 174
Tableau 21 – Primitives envoyées par la machine ARPM à la machine DMPM 177
Tableau 22 – Primitives adressées par la machine DMPM à la machine ARPM 177
Tableau 23 – Paramètres utilisés avec les primitives échangées entre la machine ARPM et la machine DMPM 178
Tableau 24 – Fonction GetArepId() 179
Tableau 25 – Fonction BuildFAS-PDU 179
Tableau 26 – Fonction FAS_Pdu_Type 179
Tableau 27 – Fonction AbortIdentifier 179
Tableau 28 – Fonction AbortReason 179
Tableau 29 – Fonction AbortDetail 179
Tableau 30 – Descriptions de l'état DMPM 180
Tableau 31 – Tableau d'états de la machine DMPM – transactions expéditeur 181
Tableau 32 – Tableau d'états de la machine DMPM – transactions destinataire 184
Tableau 33 – Primitives échangées entre la couche Liaison de données et la machine DMPM 187
Tableau 34 – Fonction PickArep 189
Tableau 35 – Fonction FindAREP 189
Tableau 36 – Fonction LocateQubArep 190
Tableau 37 – Fonction SetIdentifier() 190

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-9: Spécification du protocole de la couche application – Eléments de type 9

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

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

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

La Norme internationale CEI 61158-6-9 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette troisième édition annule et remplace la deuxième édition publiée en 2010. Cette édition constitue une révision technique. Cette édition inclut les modifications majeures par rapport à l'édition précédente:

- Correction de la plage valide de Time-difference (écart du temps)
- Correction des Passages d'états du Tableau 3
- Inclusion de la classe de ponctualité "Transparent" dans le modèle formel des points AREP BNU

Le texte de cette norme est issu des documents suivants:

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

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

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

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

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera:

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

INTRODUCTION

La présente partie de la CEI 61158 s'inscrit dans une série créée pour faciliter l'interconnexion des composants de systèmes d'automation. Elle est relative aux autres normes de l'ensemble défini par le modèle de référence de bus de terrain "à trois couches" décrit dans la CEI 61158-1.

Le protocole d'application fournit le service d'application au moyen des services disponibles au niveau de la couche Liaison de données ou de la couche immédiatement inférieure. Le principal objectif de la présente norme est de définir un ensemble de règles de communication, exprimées en termes de 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.

La présente norme porte en particulier sur la communication et l'interfonctionnement des capteurs, des effecteurs et d'autres appareils d'automatisation. Grâce à cette norme associée 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 6-9: Spécification du protocole de la couche application – Eléments de type 9

1 Domaine d'application

1.1 Généralités

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

La présente norme fournit des éléments communs pour les communications en temps critique ou non entre des programmes d'application dans un environnement et avec un matériel d'automatisation propres aux bus de terrain de Type 9. Le terme "à temps critique" sert à représenter la présence d'une fenêtre temporelle dans les limites de laquelle une ou plusieurs actions spécifiées sont exigées d'être parachevées avec un certain niveau défini de certitude. Le manquement à parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent ces actions, avec le risque concomitant pour l'équipement, les installations et éventuellement pour la vie humaine.

La présente norme définit de manière abstraite le comportement, visible par un observateur externe, assuré par la couche Application de bus de terrain de Type 9, en termes

a) de syntaxe abstraite définissant les unités de données de protocole de couche application, transmises entre les entités d'application en communication,

b) de syntaxe de transfert définissant les unités de données de protocole de couche application, transmises entre les entités d'application en communication,

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

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

La présente norme vise à définir le protocole mis en place pour

1) définir la représentation filaire des primitives de service définies dans la CEI 61158-5-9, et

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

La présente norme spécifie le protocole de la couche application de bus de terrain CEI de Type 9, conformément au modèle de référence de base OSI (ISO/CEI 7498-1) et à la structure de couche application OSI (ISO/CEI 9545).

1.2 Spécifications

La présente norme a pour principal objectif de préciser la syntaxe et les caractéristiques du protocole de couche application qui transmet les services de couche application définis dans la CEI 61158-5-9.

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles de communications industrielles préexistants. Ce dernier objectif explique la diversité des protocoles normalisés dans la CEI 61158-6.

1.3 Conformité

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

2 Références normatives

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

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

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

CEI 61158-3-1, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-1: Définition des services de la couche liaison de données – Eléments de type 1*

CEI 61158-4-1, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-1: Spécification du protocole de la couche liaison de données – Eléments de type 1*

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

CEI 61158-5-9, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-9: Définition des services de la couche application – Eléments de type 9*

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange* (disponible en anglais seulement)

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

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

ISO/IEC 8825-1, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)* (disponible en anglais seulement)

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

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

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic* (disponible en anglais seulement)

3 Termes, définitions, symboles, abréviations et conventions

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

3.1 Termes et définitions provenant d'autres normes ISO/CEI

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

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

3.1.2 Termes et définitions de l'ISO/CEI 9545

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

3.1.3 Termes et définitions provenant de l'ISO/CEI 8824-1

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

3.1.4 Termes et définitions provenant de l'ISO/CEI 8825-1

a) encodage (d'une valeur de donnée)
b) valeur de donnée
c) octets d'identification (la présente norme utilise le singulier)
d) octet(s) de longueur (la présente norme utilise à la fois le singulier et le pluriel)
e) octets de contenu

3.2 Termes de la CEI 61158-1

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

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

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

3.2.3
objet d'application
classe d'objets qui gère et assure l'échange de messages sur le réseau et dans l'appareil réseau, lors de l'exécution

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

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

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

3.2.6
objet de processus d'application
composant d'un processus d'application, identifiable et accessible via une relation d'applications de la FAL

Note 1 à l'article: Les définitions d'objets de processus d'application se composent d'un ensemble de valeurs destinées aux attributs de leur classe (voir la définition "Classe d'Objets de Processus d'Application"). On peut accéder aux définitions d'objets de processus d'application à distance à l'aide des services de l'ASE "FAL Object Management" (Gestion d'objets de la FAL). Les services de gestion des objets FAL peuvent être utilisés pour charger ou mettre à jour des définitions d'objets, pour lire des définitions d'objets, ainsi que pour créer et supprimer de manière dynamique des objets d'application et les définitions correspondantes.

3.2.7
classe d'objets de processus d'application
classe d'objets de processus d'application définis par un ensemble de services et d'attributs accessibles par le biais du réseau

3.2.8
relation entre applications
association de type coopératif entre deux invocations d'entités d'application (application-entity-invocation) ou plus, dans un but d'échange d'informations et de coordination de leur action conjointe

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

3.2.9
élément de service d'application de relation entre applications
application-service-element (élément de service d'application) qui fournit le moyen exclusif d'établir et de faire cesser toutes les relations d'applications

3.2.10
point d'extrémité de relation entre applications
contexte et comportement d'une relation d'applications, vus et maintenus par un des processus d'application impliqués dans la relation d'applications

Note 1 à l'Article: Chaque processus d'application impliqué dans la relation d'applications maintient son propre point d'extrémité de relation d'applications.

3.2.11
attribut
description d'une caractéristique ou fonction, visible par un observateur externe, d'un objet

Note 1 à l'article: Les attributs d'un objet contiennent des informations sur les portions variables d'un objet. En règle générale, ils fournissent des informations de statut ou régissent l'action d'un objet. Les attributs peuvent également influer sur le comportement d'un objet. Les attributs se répartissent en deux catégories: les attributs de classe et les attributs d'instance.

3.2.12
comportement
indication de la manière dont l'objet réagit à des événements particuliers

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

3.2.13
classe
ensemble d'objets représentant le même type de composant du système

Note 1 à l'article: Une classe est une généralisation de l'objet, un modèle qui permet de définir des variables et des méthodes. Tous les objets d'une classe possèdent une forme et un comportement identiques, mais leurs attributs contiennent généralement des données différentes.

3.2.14
attributs de classe
attribut commun à tous les objets d'une même classe

3.2.15
code de classe
identificateur non ambigu attribué à chaque classe d'objets

3.2.16
service propre à une classe
service défini par une classe d'objets particulière permettant d'exécuter une fonction exigée qu'un service commun ne peut assurer

Note 1 à l'article: Tout objet spécifique à une classe est réservé à l'usage exclusif de la classe d'objets qui le définit.

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

(b) initiateur d'un message auquel un serveur réagit; par exemple, rôle d'un point d'extrémité de relation entre applications, dans lequel ce dernier envoie des unités APDU de demande de service confirmé à un point d'extrémité de relation entre applications jouant le rôle de serveur

3.2.18
chemin de transport
flux unidirectionnel d'unités APDU à travers une relation entre applications

3.2.19
cyclique
terme utilisé pour décrire les événements qui se reproduisent de manière régulière et répétitive

3.2.20
AR dédiée
AR directement utilisée par l'utilisateur FAL

Note 1 à l'Article: Dans les AR dédiées, seuls l'en-tête de FAL et les données de l'utilisateur sont transférés

3.2.21
appareil
connexion matérielle physique à la liaison

Note 1 à l'article: Un appareil peut contenir plusieurs nœuds.

3.2.22
profil d'appareil
ensemble d'informations et de fonctionnalités dépendant de l'appareil qui garantit la cohérence entre les appareils similaires appartenant au même type d'appareil

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

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

3.2.25
erreur
divergence entre une valeur ou condition calculée, observée ou mesurée et la valeur ou condition spécifiée ou théoriquement correcte

3.2.26
classe d'erreurs
groupement général de définitions d'erreurs

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

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

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

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

3.2.29
appareil logique
classe FAL particulière qui extrait un composant logiciel ou un composant de microprogramme sous la forme d'un dispositif intégré et autonome au sein d'un appareil d'automatisation

3.2.30
informations de gestion
informations accessibles via le réseau qui facilitent la gestion de l'exploitation du système de bus de terrain, y compris la couche application

Note 1 à l'article: La gestion inclut des fonctions telles que le contrôle, la surveillance et le diagnostic.

3.2.31
réseau
série de nœuds reliés par un support de communication

Note 1 à l'article: Les chemins de connexion entre des paires de nœuds peuvent inclure des répéteurs, des routeurs et des passerelles.

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

3.2.33
point d'extrémité d'AR prédéfini
point de fin d'AR qui est défini localement dans un appareil sans utilisation du service de création

Note 1 à l'article: Les AR prédéfinies qui ne sont pas préétablies sont établies avant d'être utilisées.

3.2.34
point d'extrémité d'AR préétabli
point d'extrémité d'AR qui est mis dans un état établi pendant la configuration des AE qui contrôlent ses points d'extrémité

3.2.35
éditeur
rôle d'un point d'extrémité d'AR dans lequel il transmet des unités APDU sur le bus de terrain afin qu'elles soient consommées par un ou plusieurs abonnés

Note 1 à l'article: L'éditeur peut ne pas connaître l'identité des abonnés ou leur nombre; il peut publier ses unités APDU au moyen d'une AR dédiée. La présente norme définit deux types d'éditeurs: les éditeurs par extraction (Pull Publishers) et les éditeurs par émission (Push Publishers); chacun faisant l'objet d'une définition distincte.

3.2.36
serveur
a) rôle d'un AREP dans lequel il renvoie une unité APDU de réponse de service confirmé au client à l'origine de la demande

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

3.2.37
service
opération ou fonction qu'un objet et/ou une classe d'objets exécute à la demande d'un autre objet et/ou une autre classe d'objets

Note 1 à l'article: un ensemble de services communs est défini et des dispositions permettant la définition de services propres à un objet sont fournies. Les services propres à un objet sont des services définis par une classe d'objets particulière afin de remplir une fonction exigée qui n'est assurée par aucun service commun.

3.2.38
abonné
rôle d'un AREP dans lequel il reçoit les unités APDU produites par un éditeur

Note 1 à l'article: La présente norme définit deux types d'abonnés: les abonnés par extraction (Pull subscriber) et les abonnés par émission (Push subscriber); chacun faisant l'objet d'une définition distincte.

3.3 Abréviations et symboles

AE Application Entity (entité d'application)

AL Application Layer (couche application)

ALME Application Layer Management Entity (entité de gestion de couche application)

ALP Application Layer Protocol (protocole de couche application)

APO Application Object (objet d'application)

AP Application Process (processus d'application)

APDU Application Protocol Data Unit (unité de donnée de protocole application)

API Application Process Identifier (identificateur de processus d'application)

AR Application Relationship (relation entre applications)

AREP Application Relationship End Point (point d'extrémité de relation entre applications)

ASCII American Standard Code for Information Interchange (code normalisé américain pour l'échange d'informations)

ASE ASE Application Service Element (élément de service d'application)

Cnf Confirmation

DL- Préfixe désignant la couche Data Link (liaison de données)

DLC Data-Link Connection (connexion de couche de liaison de données)

DLCEP Data-Link Connection End Point (point de fin de connexion de couche de liaison de données)

DLL Data-link layer (couche de liaison de données)

DLM Data-Link-Management (gestion de couche de liaison de données)

DLSAP Data-Link Service Access Point (point d'accès de service de couche de liaison de données)

DLSDU DL-service-data-unit (unité de données de service de DL)

FAL Fieldbus Application Layer (couche application de bus de terrain)

ID Identificateur

CEI Commission Electrotechnique Internationale

Ind Indication

LME Layer Management Entity (entité de gestion de couche)

OSI Open Systems Interconnect (interconnexion de systèmes ouverts)

QoS Quality of Service (qualité de service)

Req Request (demande)

Rsp Response (réponse)

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

SDU Service Data Unit (unité de données de service)

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

SMK System Management Kernel (noyau de gestion système)

VFD Virtual Field Device (appareil de champ virtuel)

3.4 Conventions

3.4.1 Concept général

La couche FAL est définie comme un ensemble d'éléments ASE orientés objet. Chaque ASE est spécifié dans un paragraphe distinct. Chaque spécification d'ASE est constituée de trois

parties: ses définitions de classe, ses services et sa spécification de protocole. Les deux premiers éléments sont contenus dans la sous-série CEI 61158-5. La spécification des protocoles pour chacun des éléments ASE est définie dans la présente norme.

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

La présente norme emploie les conventions de description énoncées dans l'ISO/CEI 10731.

3.4.2 Conventions relatives aux définitions de classe

Les définitions de mapping de couche Liaison de données sont décrites au moyen de modèles. Chaque modèle est constitué d'une liste d'attributs de la classe. La forme générale des modèles est définie dans la CEI 61158-5-5.

3.4.3 Conventions de syntaxe abstraite

Lorsque le paramètre "optionalParametersMap" est utilisé, un numéro de bit correspondant à chaque production OPTIONAL ou DEFAULT est donné en commentaire.

3.5 Conventions utilisées dans les diagrammes d'états

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

Tableau 1 – Conventions utilisées dans les diagrammes d'états

#	Etat actuel	Evénement/condition => 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 les diagrammes d'états 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

Cette construction permet d'exécuter une séquence d'actions en boucle au sein d'un passage. La boucle est exécutée pour toutes les valeurs comprises entre start_value et end_value.

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

Cette construction permet d'exécuter d'autres actions dépendant d'une condition (qui peut être la valeur d'un identificateur ou le résultat d'une action antérieure) au sein d'un passage.

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

Il est fortement recommandé de se référer aux paragraphes traitant des définitions d'attributs de point AREP, des fonctions locales et des définitions d'unités de données de protocole (Protocol Data Unit, PDU) de couche FAL pour comprendre les machines de protocole. Nous supposons que les lecteurs possèdent une connaissance suffisante de ces définitions et savent les utiliser sans explications complémentaires.

4 Syntaxe abstraite

4.1 Syntaxe abstraite des unités PDU FAL-AR

4.1.1 Définition de premier niveau

Les productions définies ici doivent être utilisées avec les règles d'encodage des unités APDU (voir 5.1.2).

```
APDU::= CHOICE {
            [PRIVATE 0] ConfirmedSend-RequestPDU,
            [PRIVATE 1] ConfirmedSend-ResponsePDU,
            [PRIVATE 2] UnconfirmedSend-PDU,
            [PRIVATE 3] UnconfirmedAcknowledgedSend-CommandPDU,
            [PRIVATE 4] Establish-RequestPDU,
            [PRIVATE 5] Establish-ResponsePDU,
            [PRIVATE 6] Establish-ErrorPDU,
            [PRIVATE 7] Abort-PDU,
            [PRIVATE 8] DataSendAcknowledge-PDU
}
```

4.1.2 Service Confirmed send

```
ConfirmedSend-RequestPDU::= SEQUENCE {
      [APPLICATION 0] AddressAREP,
      InvokeID,
      ConfirmedServiceRequest
}

ConfirmedSend-ResponsePDU::= SEQUENCE {
      [APPLICATION 1] AddressAREP,
      InvokeID,
      pduBody     CHOICE {
         ConfirmedServiceResponse,
         ConfirmedServiceError
      }
}
```

4.1.3 Service Unconfirmed send

```
UnconfirmedSend-PDU::= SEQUENCE {
      [APPLICATION 2] AddressAREP,
      InvokeID,
      pduBody     CHOICE {
         UnconfirmedServiceRequest,
         UnconfirmedSendPD-PDU
      }
}
```

4.1.4 Service d'envoi d'accusé de réception non confirmé

```
UnconfirmedAcknowledgeSend-CommandPDU::= SEQUENCE {
      [APPLICATION 3] AddressAREP,
      InvokeID,
      UnconfirmedServiceRequest
}
```

4.1.5 InvokeID

```
InvokeID::= Unsigned8
```

4.1.6 Service d'établissement

```
MaxOSCC::= Unsigned8

MaxOSCS::= Unsigned8

MaxUCSC::= Unsigned8

MaxUCSS::= Unsigned8

CIU::= Unsigned32

Establish-RequestPDU::= SEQUENCE {
      [APPLICATION 4] AddressAREP,
      ConType,
      MaxOSCC,
      MaxOSCS,
      MAXUCSC,
      MAXUCSS,
      CIU,
      InvokeID,
      initiateRequest                    [PRIVATE 0] IMPLICIT Initiate-RequestPDU
}

Establish-ResponsePDU::= SEQUENCE {
      [APPLICATION 5] AddressAREP,
      InvokeID,
      initiateResponse                   [PRIVATE 0] IMPLICIT Initiate-ResponsePDU
}

Establish-ErrorPDU::= SEQUENCE {
      [APPLICATION 6] AddressAREP,
      InvokeID,
      initiateError                      [PRIVATE 0] IMPLICIT Initiate-ErrorPDU
}
```

4.1.7 ConType

```
ConType::= ENUMERATED {
   mmaz          (0)
}
```

4.1.8 Service d'accusé d'envoi de données

```
DataSendAcknowledge-PDU::= SEQUENCE {
      Protocol-Code,[APPLICATION 8],Address2ARP,      --- Protocol-Code dans le quartet supérieur de l'octet!!!
      Block-Number,
      Block-Length,
      Protocol-Data,
}
```

4.1.9 Protocol-code

```
Protocol-Code::= ENUMERATED {
   TCP/IP          (1)                    --- protocole TCP/IP
}
```

4.1.10 Block-number

```
Block-Number::= Unsigned8                 --- 0: absence de blocage
                                              ---- 1 à 254: numéro de bloc
                                              ---- 255: dernier bloc
```

4.1.11 Block-length

```
Block-Length::= Unsigned
```

4.1.12 Protocol-data

```
Protocol-Data::= Any                      --- transfert de protocole transparent
```

4.1.13 Address2 ARP

```
Address2ARP::= SEQUENCE {
   Destination-Address,
   Source-Address,
   Destination-Node,
   Destination-Subnode,
   Source-Node,
   Source-Subnode
}
```

4.1.14 Destination-address

```
Destination-Address::= OctetString                --- 3 Octets
```

4.1.15 Source-address

```
Source-Address::= OctetString                     --- 3 Octets
```

4.1.16 Destination-node

```
Destination-Node::= Unsigned8
```

4.1.17 Source-node

```
Source-Node::= Unsigned8
```

4.1.18 Destination-subnode

```
Destination-Subnode::= Unsigned8
```

4.1.19 Source-subnode

```
Source-Subnode::= Unsigned8
```

4.2 Syntaxe abstraite de PDUBody

4.2.1 Service Abort

```
Abort-PDU::= SEQUENCE {
      [APPLICATION 7] AddressAREP,
      Identifier,
      ReasonCode,
      AdditionalDetail
}
```

4.2.2 ConfirmedServiceRequest

```
ConfirmedServiceRequest::= CHOICE {
   read-Request                    [0] IMPLICIT Read-RequestPDU,
   write-Request                   [3] IMPLICIT Write-RequestPDU,
   start-Request                   [6] IMPLICIT Start-RequestPDU,
   stop-Request                    [9] IMPLICIT Stop-RequestPDU,
   status-Request                  [15] IMPLICIT Status-RequestPDU,
   identify-Request                [18] IMPLICIT Identify-RequestPDU,
   getAttributes1-Request          [21] IMPLICIT GetAttributes-RequestPDU, -- forme courte
   getAttributes2-Request          [35] IMPLICIT GetAttributes-RequestPDU, -- forme longue
   reset-Request                   [36] IMPLICIT Reset-RequestPDU,
   resume-Request                  [39] IMPLICIT Resume-RequestPDU
}
```

4.2.3 ConfirmedServiceResponse

```
ConfirmedServiceResponse::= CHOICE {
   read-Response                       [1] IMPLICIT Read-ResponsePDU,
   write-Response                      [4] IMPLICIT Write-ResponsePDU,
   start-Response                      [7] IMPLICIT Start-ResponsePDU,
   stop-Response                   [10] IMPLICIT Stop-ResponsePDU,
   status-Response                 [16] IMPLICIT Status-ResponsePDU,
   identify-Response               [19] IMPLICIT Identify-ResponsePDU,
   getAttributes-Response          [22] IMPLICIT GetAttributes-ResponsePDU,
   reset-Response                      [37] IMPLICIT Reset-ResponsePDU,
   resume-Response                 [40] IMPLICIT Resume-ResponsePDU
}
```

4.2.4 ConfirmedServiceError

```
ConfirmedServiceError::= CHOICE {
   read-Error                      [2] IMPLICIT ErrorType,
   write-Error                     [5] IMPLICIT ErrorType,
   start-Error                     [8] IMPLICIT ErrorFiType,
   stop-Error                      [11] IMPLICIT ErrorFiType,
   status-Error                    [17] IMPLICIT ErrorType,
   identify-Error                  [20] IMPLICIT ErrorType,
   getAttributes-Error             [23] IMPLICIT ErrorType,
   reset-Error                     [38] IMPLICIT ErrorFiType,
   resume-Error                    [41] IMPLICIT ErrorFiType
}
```

4.2.5 Type d'erreur

Type d'erreur selon les spécifications de 4.1.4.6.

4.2.6 Error Fi type

```
ErrorFiType::= SEQUENCE {
   errorClass                      [0] IMPLICIT ErrorClass,
   additionalCode                  [1] IMPLICIT Integer16 OPTIONAL,
   fiState                         [3] IMPLICIT Integer8
}
```

4.2.7 Classe d'erreurs

Classe d'erreurs selon les spécifications de 4.1.4.7.

4.2.8 Unités PDU non confirmées

```
UnconfirmedServiceRequest::= CHOICE {
   informationReport-Request       [12] IMPLICIT InformationReport-RequestPDU,
   reject-Request                  [34] IMPLICIT Reject-RequestPDU
}

UnconfirmedSendPD-PDU::= BIT STRING
```

4.2.9 Elément ASE de gestion

4.2.9.1 Service d'obtention d'attributs

```
GetAttributes-Request-PDU::= SEQUENCE {
   listOfAttributes                    [PRIVATE 0] IMPLICIT Mn_SelectedAttributes,
   accessSpecification CHOICE {
      index                               [1] IMPLICIT Gn_NumericID,
      variableName                     [2] IMPLICIT Gn_Name,
      fiName                           [5] IMPLICIT Gn_Name,
      startIndex                       [7] IMPLICIT Gn_NumericID
   }
}

GetAttributes-ResponsePDU::= SEQUENCE {
   more                                [PRIVATE 0] IMPLICIT Gn_MoreFollows,
   listOfObjectDefinition              [PRIVATE 1] IMPLICIT SEQUENCE OF Gn_ObjectDefinition
}
```

4.2.10 Elément ASE de processus d'application

4.2.10.1 Service d'obtention d'état

```
Status-RequestPDU::= NULL

Status-ResponsePDU::= SEQUENCE {
   logicalStatus                       [PRIVATE 0] IMPLICIT ENUMERATED {
      ready-for-communication             (0),
      limited-services-permitted          (2),
   },
   physicalStatus                      [PRIVATE 1] IMPLICIT ENUMERATED {
      operational                         (0),
      partially-operational               (1),
      inoperable                          (2),
      needs-commissioning                 (3)
   },
   localDetail                         [PRIVATE 2] IMPLICIT BitString OPTIONAL
 }
```

4.2.10.2 Identify service

```
Identify-RequestPDU::= NULL

Identify-ResponsePDU::= SEQUENCE {
   vendorName                          [PRIVATE 0] IMPLICIT VisibleString,
   modelIdentifier                     [PRIVATE 1] IMPLICIT VisibleString,
   vendorRevision                         [PRIVATE 2] IMPLICIT VisibleString
}
```

4.2.10.3 Service Initiate

```
Initiate-RequestPDU::= SEQUENCE {
   versionObjectDefinitionsCalling     [PRIVATE 0] IMPLICIT Integer16,
   apDescriptorCalling                 [PRIVATE 1] IMPLICIT OctetString,
   accessProtectionSupportedCalling    [PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
   passwordAndAccessGroupsCalling      [PRIVATE 3] IMPLICIT Ap_AccessControl,
   configuredMaxPduSizeSending         [PRIVATE 4] IMPLICIT Unsigned8,
   configuredMaxPduSizeReceiving       [PRIVATE 5] IMPLICIT Unsigned8,
   listOfSupportedServicesCalling      [PRIVATE 6] IMPLICIT Mn_PduSupportedMap
}

Initiate-ResponsePDU::= SEQUENCE {
   versionObjectDefinitionsCalled      [PRIVATE 0] IMPLICIT Integer16,
   apDescriptorCalled                  [PRIVATE 1] IMPLICIT OctetString,
   accessPrivilegeSupportedCalled      [PRIVATE 2] IMPLICIT Ap_AccessProtectionSupported,
   passwordAndAccessGroupsCalled       [PRIVATE 3] IMPLICIT Ap_AccessControl
}
```

```
Initiate-ErrorPDU::= SEQUENCE {
   errorCode                        [PRIVATE 0] IMPLICIT ENUMERATED {
      other                            (0),
      max-fal-pdu-size-insufficient    (1),
      service-not-supported            (2),
      version-obj-def-incompatible     (3),
      user-initiate-denied             (4),
      password-error                   (5),
      profile-number-incompatible      (6)
      },
   maxPduLengthSendingCalled        [PRIVATE 1] IMPLICIT Unsigned8,
   maxPduLengthReceivingCalled      [PRIVATE 2] IMPLICIT Unsigned8,
   listOfSupportedServicesCalled    [PRIVATE 3] IMPLICIT Mn_PduSupportedMap
}
```

4.2.10.4 Service Reject

```
Reject-RequestPDU::= SEQUENCE {
   original-invokeID                [PRIVATE 0] IMPLICIT Integer8,
   reject-code                      [PRIVATE 1] IMPLICIT ENUMERATED {
      pdu-size                         (5)
   }
}
```

4.2.11 Elément ASE d'invocation de fonctions

4.2.11.1 Service Reset

```
Reset-RequestPDU::= SEQUENCE {
   keyAttribute                     Gn_KeyAttribute
}

Reset-ResponsePDU::= NULL
```

4.2.11.2 Service Resume

```
Resume-RequestPDU::= SEQUENCE {
   keyAttribute                     Gn_KeyAttribute
}

Resume-ResponsePDU::= NULL
```

4.2.11.3 Service Start

```
Start-RequestPDU::= SEQUENCE {
   keyAttribute                     Gn_KeyAttribute
}

Start-ResponsePDU::= NULL
```

4.2.11.4 Service Stop

```
Stop-RequestPDU::= SEQUENCE {
   keyAttribute                     Gn_KeyAttribute
}

Stop-ResponsePDU::= NULL
```

4.2.12 ASE Variable

4.2.12.1 Information report service

```
InformationReport-RequestPDU::= SEQUENCE {
   index                            Gn_NumericID,
   subIndex                         [PRIVATE 0] IMPLICIT Gn_SubIndex OPTIONAL,
   value                            [PRIVATE 1] IMPLICIT ANY
}
```

4.2.12.2 Service Read

```
Read-RequestPDU::= SEQUENCE {
   index                            Gn_NumericID,
   subIndex                         [PRIVATE 0] IMPLICIT Gn_SubIndex OPTIONAL
}

Read-ResponsePDU::= SEQUENCE {
   value                                [PRIVATE 0] IMPLICIT ANY
}
```

4.2.12.3 Write service

```
Write-RequestPDU::= SEQUENCE {
   index                                Gn_NumericID,
   subIndex                             Gn_SubIndex OPTIONAL,
   value                                [PRIVATE 0] IMPLICIT ANY
}

Write-ResponsePDU::= NULL
```

4.3 Définitions de types d'éléments ASE

4.3.1 Types d'éléments ASE AP

4.3.1.1 Ap_AccessProtectionSupported

```
Ap_AccessProtectionSupported::= Boolean     -- True signifie que la protection d'accès est prise en charge.
                                            -- False signifie que la protection d'accès n'est pas prise en charge.
```

4.3.1.2 Ap_AccessControl

```
Ap_AccessControl::= BitString {             -- Le mot de passe (password – Unsigned8) est encodé sous la forme
d'une chaîne de bits.
   password_Bit1                        (8),
   password_Bit2                        (7),
   password_Bit3                        (6),
   password_Bit4                        (5),
   password_Bit5                        (4),
   password_Bit6                        (3),
   password_Bit7                        (2),
   password_Bit8                        (1),
   access_Groups-1                      (16),
   access_Groups-2                      (15),
   access_Groups-3                      (14),
   access_Groups-4                      (13),
   access_Groups-5                      (12),
   access_Groups-6                      (11),
   access_Groups-7                      (10),
   access_Groups-8                      (9)
}
```

4.3.2 Types d'éléments ASE AR

4.3.2.1 Code de motif et informations supplémentaires

4.3.2.1.1 Code de motif

```
ReasonCode::= Unsigned8
```

4.3.2.1.2 Informations supplémentaires

```
AdditionalDetailI::= OctetString
```

4.3.2.2 AREP

```
AddressAREP::= Unsigned8                    -- Octet de poids le plus faible de l'adresse du point DLCEP
```

4.3.2.3 Identificateur de type de coupure

```
Identifier::= ENUMERATED {
   fal-user                             (0),
   fal-APO-ASE                          (1),
   fal-AR-ASE                           (2),
   dll                                  (3)
}
```

4.3.3 Types d'éléments ASE d'invocation de fonctions

4.3.3.1 Fi_AccessPrivilege

```
Fi_AccessPrivilege::= BitString {
    rightToStartPassword            (24),
    rightToStopPassword             (23),
    rightToDeletePassword           (22),
    rightToStartAccessGroup         (20),
    rightToStopAccessGroup          (19),
    rightToDeleteAccessGroup        (18),
    rightToStartAllPartner          (32),
    rightToStopAllPartner           (31),
    rightToDeleteAllPartner         (30),
    password_Bit1                   (8),    -- Le mot de passe (password – Unsigned8) est encodé sous la
forme d'une chaîne de bits.
    password_Bit2                   (7),
    password_Bit3                   (6),
    password_Bit4                   (5),
    password_Bit5                   (4),
    password_Bit6                   (3),
    password_Bit7                   (2),
    password_Bit8                   (1),
    access_Groups-1                 (16),
    access_Groups-2                 (15),
    access_Groups-3                 (14),
    access_Groups-4                 (13),
    access_Groups-5                 (12),
    access_Groups-6                 (11),
    access_Groups-7                 (10),
    access_Groups-8                 (9)
}
```

4.3.3.2 Fi_State

```
Fi_State::= Unsigned8 {
    unrunnable                      (1),
    idle                            (2),
    running                         (3),
    stopped                         (4),
    starting                        (5),
    stopping                        (6),
    resuming                        (7),
    resetting                       (8)
}
```

4.3.4 Types d'éléments ASE de gestion

4.3.4.1 Mn_PduSupportedMap

```
Mn_PduSupportedMap::= BIT STRING {
    getAttributes-RequestPDU        (1),        -- Demandeur
    start-RequestPDU                (8),
    stop-RequestPDU                 (9),
    resume-RequestPDU               (9),
    reset-RequestPDU                (9),
    read-RequestPDU                 (11),
    write-RequestPDU                (12),
    informationReport-RequestPDU    (17),
    getAttributes-ResponsePDU       (25),       -- Répondeur
    start-ResponsePDU               (33),
    stop-ResponsePDU                (33),
    resume-ResponsePDU              (33),
    reset-ResponsePDU               (33),
    read-ResponsePDU                (35),
    write-ResponsePDU               (36),
    informationReport-ResponsePDU   (41)
}
```

4.3.5 Types d'éléments ASE de variable

4.3.5.1 Vr_AccessPrivilege

```
Vr_AccessPrivilege::= BitString {
   rightToReadPassword              (24),
   rightToWritePassword             (23),
   rightToDeletePassword            (22),
   rightToReadAccessGroup           (20),
   rightToWriteAccessGroup          (19),
   rightToDeleteAccessGroup         (18),
   rightToReadAllPartners           (32),
   rightToWriteAllPartners          (31),
   rightToDeleteAllPartners         (30),
   password_Bit1                    (8),        -- Le mot de passe (password – Unsigned8) est encodé sous
la forme d'une chaîne de bits.
   password_Bit2                    (7),
   password_Bit3                    (6),
   password_Bit4                    (5),
   password_Bit5                    (4),
   password_Bit6                    (3),
   password_Bit7                    (2),
   password_Bit8                    (1),
   access_Groups-1                  (16),
   access_Groups-2                  (15),
   access_Groups-3                  (14),
   access_Groups-4                  (13),
   access_Groups-5                  (12),
   access_Groups-6                  (11),
   access_Groups-7                  (10),
   access_Groups-8                  (9)
}
```

4.3.6 Types généraux

4.3.6.1 Gn_Deletable

```
Gn_Deletable::= Boolean             -- True signifie peut être supprimé.
                                    -- False signifie ne peut pas être supprimé.
```

4.3.6.2 Gn_KeyAttribute

```
Gn_KeyAttribute::= CHOICE {
-- Lorsque ce type est spécifié, seuls les attributs clés de la classe référencée sont valides.
   numericID                        [0] IMPLICIT Gn_NumericID,
   name                             [1] IMPLICIT Gn_Name,
   listName                         [2] IMPLICIT Gn_Name,
   numericAddress                   [4] IMPLICIT Gn_NumericAddress,
   symbolicAddress                  [5] IMPLICIT Gn_SymbolicAddress
}
```

4.3.6.3 Gn_Length

```
Gn_Length::= Unsigned8
```

4.3.6.4 Gn_MoreFollows

```
Gn_MoreFollows::= Boolean
```

4.3.6.5 Gn_NumericID

```
Gn_NumericID::= Unsigned16          -- Les valeurs de ce paramètre sont uniques au sein d'un AP.
```

4.3.6.6 Gn_Name

```
Gn_Name::= OctetString
```

4.3.6.7 Gn_ObjectClass

```
Gn_ObjectClass::= ENUMERATED {
   functionInvocation                (3),
   fixedLengthStringDataType         (5),
   structuredDataType                (6),
   fixedLengthStringVariable         (7),
   arrayVariable                     (8),
   dataStructureVariable             (9),
}
```

4.3.6.8 Gn_ObjectDefinition

```
Gn_ObjectDefinition::= OctetString        -- La sémantique de ce paramètre est propre à l'application.
```

4.3.6.9 Gn_Reusable

```
Gn_Reusable::= Boolean                    -- True signifie réutilisable.
                                          -- False signifie non réutilisable.
```

4.3.6.10 Gn_SubIndex

```
Gn_SubIndex::= Unsigned8
```

4.3.6.11 Gn_TypeDescription

```
Gn_TypeDescription::= CHOICE {
   boolean                              [1] Gn_Length,
   integer8                             [2] Gn_Length,
   integer16                         [3] Gn_Length,
   integer32                         [4] Gn_Length,
   unsigned8                         [5] Gn_Length,
   unsigned16                        [6] Gn_Length,
   unsigned32                        [7] Gn_Length,
   float                             [8] Gn_Length,
   visiblestring                     [9] Gn_Length,
   octetstring                       [10] Gn_Length,
   binaryDate                        [11] Gn_Length,
   timeOfDay                         [12] Gn_Length,
   timeDifference                    [13] Gn_Length,
   bitstring                            [14] Gn_Length
}
```

4.3.7 Définitions d'objets

4.3.7.1 Définition de premier niveau

```
Object-Definition::= CHOICE {
            [PRIVATE 0] ListHeader,
            [PRIVATE 1] DataTypeList,
            [PRIVATE 2] StaticList,
            [PRIVATE 3] FunctionInvocationDefinition
}
```

4.3.7.2 ListHeader

```
ListHeader::= SEQUENCE {
   numericId                                  [PRIVATE 0] IMPLICIT Unsigned16,
   romRamFlag                                 [PRIVATE 1] IMPLICIT Boolean,
   maxNameLength                              [PRIVATE 2] IMPLICIT Unsigned8,
   accessProtectionSupported                  [PRIVATE 3] IMPLICIT Boolean,
   versionOfObjectDefinition                  [PRIVATE 4] IMPLICIT Integer16,
   localReferenceOfListHeader                 [PRIVATE 5] IMPLICIT Unsigned32 OPTIONAL,
   numberOfEntriesInDataTypeList              [PRIVATE 6] IMPLICIT Unsigned16,
   localReferenceOfDataTypeList               [PRIVATE 7] IMPLICIT Unsigned32 OPTIONAL,
   firstNumericIdOfStaticList                 [PRIVATE 8] IMPLICIT Unsigned16,
   numberOfEntriesInStaticList                [PRIVATE 9] IMPLICIT Unsigned16,
   localReferenceOfStaticList                 [PRIVATE 10] IMPLICIT Unsigned32 OPTIONAL,
   firstNumericIdOfVariableListDefinition     [PRIVATE 11] IMPLICIT Unsigned16,
   numberOfEntriesInVariableListDefinition        [PRIVATE 12] IMPLICIT Unsigned16,
   localReferenceOfVariableListDefinition     [PRIVATE 13] IMPLICIT Unsigned32 OPTIONAL,
   firstNumericIdOfFunctionInvocationDefinition [PRIVATE 14] IMPLICIT Unsigned16,
   numberOfEntriesInFunctionInvocationDefinition [PRIVATE 15] IMPLICIT Unsigned16,
   localReferenceOfFunctionInvocationDefinition [PRIVATE 16] IMPLICIT Unsigned32 OPTIONAL
}
```

4.3.7.3 DataTypeList

```
DataTypeList::= CHOICE {
        [PRIVATE 0] DataTypeDefinition,
        [PRIVATE 1] StructuredDataTypeDefinition
}

DataTypeDefinition::= SEQUENCE {
   numericId                              Gn_NumericID,
   objectClass                            Gn_ObjectClass,
   dataTypeNameLength                     Gn_Length,
   dataTypeName                           [PRIVATE 0] IMPLICIT VisibleString OPTIONAL
}

StructuredDataTypeDefinition::= SEQUENCE {
   numericId                              Gn_NumericID,
   objectClass                            Gn_ObjectClass,
   numberOfElements                    [PRIVATE 0] IMPLICIT Integer8,
   recordList      SEQUENCE OF SEQUENCE{
      numericIdOfDataTypeDefinition       Gn_NumericID,
      dataLength                          Gn_Length
   }
}
```

4.3.7.4 StaticList

```
StaticList::= CHOICE {
        [PRIVATE 0] VariableDefinition,
        [PRIVATE 1] ArrayDefinition,
        [PRIVATE 2] StructureDefinition
}

VariableDefinition::= SEQUENCE {
   numericId                              Gn_NumericID,
   objectClass                            Gn_ObjectClass,
   numericIdOfDataTypeDefinition          Gn_NumericID,
   dataLength                             Gn_Length,
   accessPrivilege                        Vr_AccessPrivilege OPTIONAL,
   localReferenceOfVariable            [PRIVATE 0] IMPLICIT Unsigned32 OPTIONAL,
   variableName                        [PRIVATE 1] IMPLICIT VisibleString OPTIONAL,
   extension                           [PRIVATE 2] IMPLICIT OctetString OPTIONAL
}

ArrayDefinition::= SEQUENCE {
   numericId                              Gn_NumericID,
   objectClass                            Gn_ObjectClass,
   numericIdOfDataTypeDefinition          Gn_NumericID,
   dataLength                             Gn_Length,
   numberOfElements                    [PRIVATE 0] IMPLICIT Unsigned8,
   accessPrivilege                        Vr_AccessPrivilege OPTIONAL,
   localReferenceOfArray               [PRIVATE 1] IMPLICIT Unsigned32 OPTIONAL,
   arrayName                           [PRIVATE 2] IMPLICIT VisibleString OPTIONAL,
   extension                           [PRIVATE 3] IMPLICIT OctetString OPTIONAL
}

StructureDefinition::= SEQUENCE {
   numericId                              Gn_NumericID,
   objectClass                            Gn_ObjectClass,
   numericIdOfDataTypeDefinition          Gn_NumericID,
   accessPrivilege                        Vr_AccessPrivilege OPTIONAL,
   structureName                       [PRIVATE 0] IMPLICIT VisibleString OPTIONAL,
   extension                           [PRIVATE 1] IMPLICIT OctetString OPTIONAL,
   localReferenceOfElement             [PRIVATE 2] IMPLICIT SEQUENCE OF Unsigned32 OPTIONAL
}
```

4.3.7.5 FunctionInvocationDefinition

```
FunctionInvocationDefinition::= SEQUENCE {
   numericId                          Gn_NumericID,
   objectClass                        Gn_ObjectClass,
   numberOfRelatedObjects             [PRIVATE 0] IMPLICIT Unsigned8,
   accessPrivilege                    Fi_AccessPrivilege OPTIONAL,
   deletable                          Gn_Deletable,
   reusable                           Gn_Reusable,
   functionInvocationState            FI_State,
   numericIdOfLoadRegion              SEQUENCE OF Gn_NumericID,
   functionInvocationName             [PRIVATE 1] IMPLICIT VisibleString OPTIONAL,
   extension                          [PRIVATE 2] IMPLICIT OctetString OPTIONAL
}
```

4.4 Syntaxe abstraite des data types

4.4.1 Data types de référence

4.4.2 Notation du type Boolean

```
Boolean::= BOOLEAN                  -- TRUE si la valeur est non nulle.
                                    -- FALSE si la valeur est nulle.
```

4.4.3 Notation des types Integer

```
Integer::= INTEGER                  -- n'importe quel entier
Integer8::= INTEGER (-128..+127)    -- plage -27 <= i <= 27-1
Integer16::= INTEGER (-32768..+32767) -- plage -215 <= i <= 215-1
Integer32::= INTEGER                -- plage -231 <= i <= 231-1
```

4.4.4 Notation des types Unsigned

```
Unsigned::= INTEGER                 -- n'importe quel entier naturel
Unsigned8::= INTEGER (0..255)       -- plage 0 <= i <= 28-1
Unsigned16::= INTEGER (0..65535)    -- plage 0 <= i <= 216-1
Unsigned32::= INTEGER               -- plage 0 <= i <= 232-1
```

4.4.5 Notation du type Floating Point

```
Floating32::= BIT STRING SIZE (4)   -- ISO/CEI/IEEE 60559 Simple précision
```

4.4.6 Notation du type BitString

```
BitString::= BIT STRING             -- Utilisation générale
BitString8::= BIT STRING SIZE (8)   -- Chaîne de bits fixe de huit bits
BitString16::= BIT STRING SIZE (16) -- Chaîne de bits fixe de 16 bits
BitString32::= BIT STRING SIZE (32) -- Chaîne de bits fixe de 32 bits
```

4.4.7 Notation du type OctetString

```
OctetString::= OCTET STRING             --Utilisation générale
OctetString2::= OCTET STRING SIZE (2)   -- Chaîne d'octets fixe de deux octets
OctetString4::= OCTET STRING SIZE (4)   -- Chaîne d'octets fixe de quatre octets
OctetString6::= OCTET STRING SIZE (6)   -- Chaîne d'octets fixe de six octets
OctetString7::= OCTET STRING SIZE (7)   -- Chaîne d'octets fixe de sept octets
OctetString8::= OCTET STRING SIZE (8)   -- Chaîne d'octets fixe de huit octets
OctetString16::= OCTET STRING SIZE (16) -- Chaîne d'octets fixe de seize octets
```

4.4.8 Notation du type VisibleString

```
VisibleString2::= VisibleString SIZE (2)    -- Chaîne visible fixe de deux octets
VisibleString4::=VisibleString SIZE (4)     -- Chaîne visible fixe de quatre octets
VisibleString8::= VisibleString SIZE (8)    -- Chaîne visible fixe de huit octets
VisibleString16::= VisibleString SIZE (16)  -- Chaîne visible fixe de seize octets
```

4.4.9 Notation du type BinaryDate

```
BinaryDate::= OctetString7
```

4.4.10 Notation du type TimeOfDay

```
TimeOfDay::= OctetString6
```

4.4.11 Notation du type TimeDifference

```
TimeDifference::= OctetString6
```

4.4.12 Notation du type TimeValue

TimeValue::= OctetString8

4.4.13 Notation du type DL—Time-offset

DL-Time-offsetType::= OctetString

5 Syntaxe de transfert

5.1.1 Généralités

Des informations supplémentaires doivent être ajoutées aux données d'utilisateur pour permettre une association non ambiguë des données au niveau du partenaire de communication. Les règles de codage des informations supplémentaires sont optimisées pour produire des messages aussi courts que possible, conformément aux exigences du bus de terrain. La fréquence d'apparition des messages spéciaux est prise en compte.

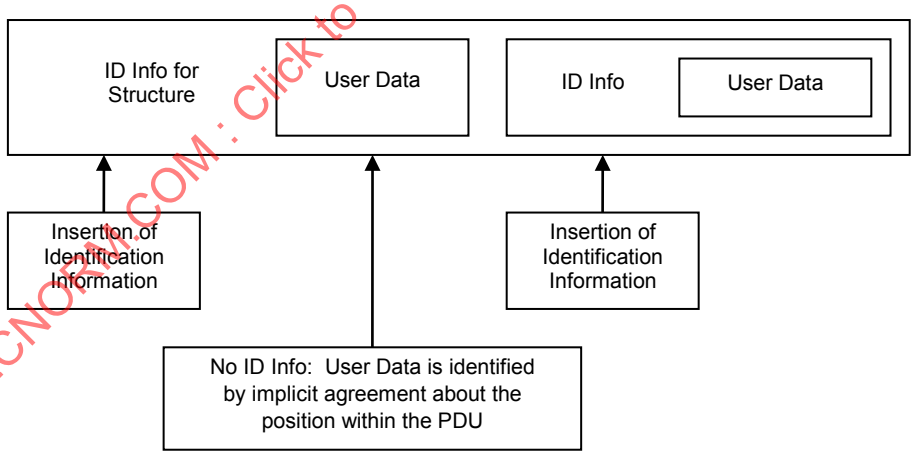
Les conditions dans la zone du bus de terrain sont les suivantes:

- messages courts
- petit nombre de messages différents
- apparition particulièrement fréquente de certains messages tels que les messages de lecture et d'écriture.

5.1.2 Règles de codage

La structuration d'une unité PDU FMS (Fieldbus Message Specification) s'effectue soit par insertion explicite d'une structure d'informations d'identification, soit par accords tacites.

La structure d'une unité PDU est représentée à la Figure 1.



Légende

Anglais	Français
ID Info for Structure	ID Info destiné à la structure
User Data	Données d'utilisateur
Insertion of Identification Information	Insertion des informations d'identification
No ID Info: User Data is identified by implicit agreement about the position within the PDU	Pas d'ID Info: données d'utilisateur identifiées par accord tacite sur la position dans l'unité PDU

Figure 1 – Insertion d'informations d'identification dans l'unité PDU FMS

Les informations d'identification se composent de l'indicateur P/C, du marqueur et de la longueur. Les structures et les données d'utilisateur de l'unité PDU peuvent être identifiées au moyen de ces informations d'identification.

La sémantique des données d'utilisateur est soit déterminée à partir du contexte, soit universellement connue (marqueurs propres au contexte ou marqueurs universels). Dans la description de la syntaxe, les marqueurs propres au contexte sont placés entre crochets. Si la sémantique d'un paramètre est déterminée de manière implicite à partir de sa position dans l'unité PDU, aucun marqueur n'est utilisé.

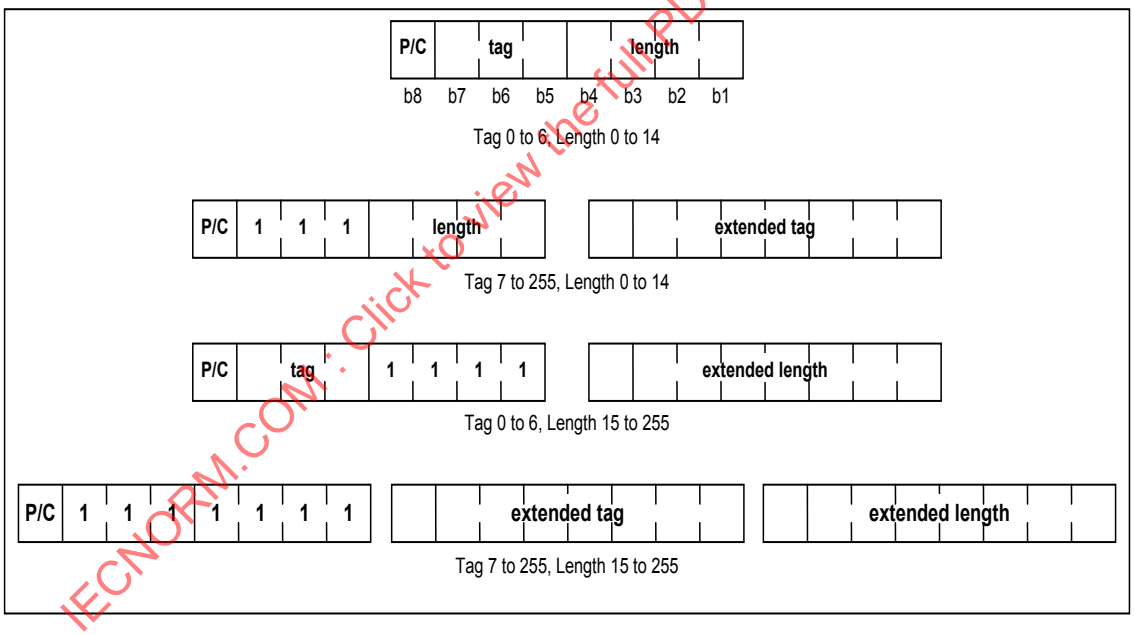
Les restrictions suivantes, qui portent sur l'emploi des composants déterminés de manière implicite (marqueurs universels), doivent être établies:

- la longueur des données d'utilisateur doit être fixe,
- le composant peut ne pas être OPTIONAL,
- le composant peut ne pas se trouver à l'intérieur d'une construction CHOICE.

5.1.3 Structure des informations d'identification

5.1.3.1 Généralités

Les informations d'identification (ID Info) se composent de l'indicateur P/C, du marqueur et de la longueur (Figure 2).



Légende

Anglais	Français
P/C	P/C
tag	repère
length	longueur
Tag 0 to 6, Length 0 to 14	Repère 0 à 6, longueur 0 à 14
extended tag	repère étendu
Tag 7 to 255, Length 0 to 14	Repère 7 à 255, longueur 0 à 14
extended length	longueur étendue
Tag 0 to 6, Length 15 to 255	Repère 0 à 6, longueur 15 à 255
extended length	longueur étendue
Tag 7 to 255, Length 15 to 255	Repère 7 à 255, longueur 15 à 255

Figure 2 – Identification

L'indicateur P/C indique s'il s'agit d'un composant simple (primitive) ou d'un composant structuré (construit, SEQUENCE, SEQUENCE OF).

Indicateur P/C = 0 <=> composant simple

Indicateur P/C = 1 <=> composant structuré

Le marqueur identifie la sémantique du composant.

La longueur correspond à la longueur en octets du composant s'il s'agit d'un composant simple et au nombre de composants contenus s'il s'agit d'un composant structuré.

La totalité de l'ID Info est codée sur un octet, si possible. L'octet est divisé en 3 parties de longueurs différentes.

- indicateur P/C 1 bit
- marqueur 3 bits
- longueur 4 bits

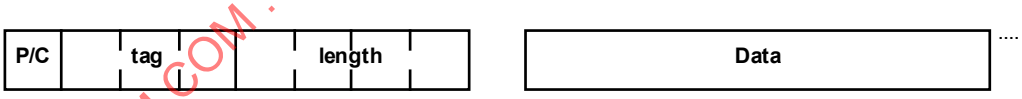
Si l'espace affecté aux champs de longueur et de marqueur n'est pas suffisant, une extension est utilisée. Pour cela, tous les bits du champ concerné sont définis sur un. Les informations sont ensuite encodées sur l'octet suivant. La plage du marqueur s'étend de 0 à 6 et celle de la longueur, de 0 à 14 lorsque aucune extension n'est utilisée. Ces plages sont préférentielles pour les messages les plus fréquents car elles produisent des unités PDU très courtes.

Si une extension est requise à la fois pour le marqueur et pour la longueur, le repère est encodé sur l'octet placé immédiatement après et la longueur, sur l'octet suivant ce dernier.

5.1.3.2 Types de données

5.1.3.2.1 Généralités

Les données d'utilisateur constituent toujours un composant simple (primitive). Elles sont encodées comme indiqué à la Figure 3.



Légende

Anglais	Français
P/C	P/C
tag	repère
length	longueur
Data	Données

Figure 3 – Codage avec identification

Si la sémantique des données d'utilisateur est déterminée de manière implicite à partir de la position au sein de l'unité PDU et que la longueur est fixe et déterminée de manière implicite, aucune information d'identification n'est ajoutée (voir Figure 4).

Data

....

Légende

Anglais	Français
Data	Données

Figure 4 – Codage sans identification

5.1.3.2.2 Codage du type Boolean

La valeur true ou false est représentée sur un octet comme indiqué à la Figure 5 et à la Figure 6.

Notation: Plage de valeurs:					**Booléen**						
Codage:					true ou false false est représenté par la valeur 0, true par la valeur FF						
	bits	8	7	6	5	4	3	2	1		
	octet 1	1	1	1	1	1	1	1	1		

Figure 5 – Représentation de la valeur true

	bits	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**		
	octet 1	0	0	0	0	0	0	0	0		

Figure 6 – Représentation de la valeur false

5.1.3.2.3 Codage des types Integer

Les valeurs Integer sont des quantités algébriques (voir Figure 7).

Notation:	**Integer8, Integer16, Integer32**		
Plage de valeurs:	Type de données	plage de valeurs	longueur
	Integer8	$-128 \le i \le 127$	1 octet
	Integer16	$-32768 \le i \le 32767$	2 octets
	Integer32	$-2^{31} \le i \le 2^{31} - 1$	4 octets
Codage:	Dans la représentation par complément à deux le bit de poids le plus fort (Most Significant Bit, MSB) est le bit qui suit le bit de signe (sign bit, SN) dans le premier octet. SN = 0: nombres positifs et z SN = 1: nombres négatifs		

	bits	8	7	6	5	4	3	2	1
	octets 1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
	2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Figure 7 – Codage des données de type Integer16

5.1.3.2.4 Codage des types Unsigned

Les valeurs Unsigned sont encodées comme indiqué à la Figure 8.

Notation:	**Unsigned8, Unsigned16, Unsigned32**		
Plage de valeurs:	**Type de données**	**plage de valeurs**	**longueur**
	Unsigned8 Unsigned16 Unsigned32	0 ≤ i ≤ 255 0 ≤ i ≤ 65 535 0 ≤ i ≤ 4 294 967 295	1 octet 2 octets 4 octets

Codage	**Binaire**							
bits	8	7	6	5	4	3	2	1
octets								
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Figure 8 – Codage des données de type Unsigned16

5.1.3.2.5 Codage du type Floating Point

Les valeurs de type Floating Point sont encodées comme indiqué à la Figure 9.

Notation:	**Floating-Point (4 octets)**
Plage de valeurs:	voir IEEE 754, nombre réel court (32 bits)
Codage:	voir IEEE 754, nombre réel court (32 bits)

bits	**8**	**7**	**6**	**5**	**4**	**3**	**2**	LSB **1**
octets		**Exposant (E)**						
1	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
	(E)	**Fraction (F)**						
2	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
	Fraction (F)							
3	2^{-8}	2^{-9}	2-10	2-11	2-12	2-13	2-14	2-15
	Fraction (F)							
4	2-16	2-17	2-18	2-19	2-20	2-21	2-22	2^{-23}

SN: signe 0 = positif, 1 = négatif

Figure 9 – Codage des données de type Floating Point

5.1.3.2.6 Codage du type Visible String

Les valeurs du type Visible String sont encodées comme indiqué à la Figure 10.

Notation:	**Visible-String**
Plage de valeurs:	voir ISO/CEI 646 et ISO/CEI 2375: définition
enregistrement	
	nombre 2 + ESPACE
Codage:	voir ISO/CEI 646

bits	8	7	6	5	4	3	2	1
octets								
1	premier caractère							
2	deuxième caractère							
.	etc.							
.								
n								

Figure 10 – Codage des données de type Visible String

5.1.3.2.7 Codage du type Octet String

Les valeurs de type Octet String sont encodées comme indiqué à la Figure 11.

Notation: Octet String
Codage: Binaire

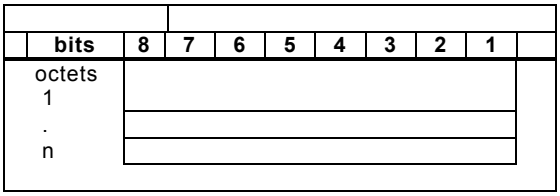


Figure 11 – Codage des données de type Octet String

5.1.3.2.8 Codage du type Date

Le data type Date se compose d'une date calendaire et d'une heure (voir Tableau 2 et Figure 12).

Notation: Date/Time
Plage de valeurs: ms à 99 ans
Codage: sur 7 octets

Tableau 2 – Codage du type Date

Paramètre	Plage de valeurs	Signification des paramètres
ms	0…59 999	millisecondes
min	0…59	nombre de minutes
SU	0,1	0: heure normale, 1: heure d'été
RSV	—	réservé
h	0…23	nombre d'heures
d. of w.	1…7	jour de la semaine: 1 = lundi, 7 = dimanche
d. of m.	1…31	jour du mois
nombre de mois	1…12	nombre de mois
years	0…99	nombre d'années (sans indication de siècle)

bits	8	7	6	5	4	3	2	1	
octets 1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 999 ms
3	RSV	RSV	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 min
4	SU	RSV	RSV	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…23 h
5	jour de la semaine 2^{2}	2^{1}	2^{0}	jour du mois 2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…7 j. de la s. 1…31 j. du m.
6	RSV	RSV	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…12 mois
7	RSV	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0 … 99 ans
	msb								

Figure 12 – Codage des données de type Date

5.1.3.2.9 Codage du type Time-of-day

Le data type Time-of-day se compose d'une heure et d'une date facultative.

L'heure correspond au nombre de millisecondes écoulées depuis minuit. La valeur est réinitialisée à minuit.

La date est indiquée en jours par rapport au 1er janvier 1984. Le premier jour de janvier 1984, la date commence à la valeur zéro.

Notation: Time-of-day

Plage de valeurs: $0 \le i \le (2^{28} -1)$ ms
$0 \le i \le (2^{16} -1)$ jours

Codage:

L'heure est représentée par une valeur binaire de 32 bits. Les quatre premiers bits (MSB) doivent avoir la valeur zéro.

La date (facultative) est encodée sous la forme d'une valeur binaire de 16 bits (2 octets).

Time-of-day est une chaîne de 4 ou 6 octets, comme indiqué à la Figure 13.

bits	8	7	6	5	4	3	2	1	
octets 1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	nombre de
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	millisecondes écoulées depuis
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	minuit
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	nombre de jours écoulés depuis le 1984-01-01
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	facultatif
	msb								

Figure 13 – Codage des données de type Time-of-day

5.1.3.2.10 Codage du type Time-difference

Le data type Time-difference se compose d'une heure en millisecondes et d'un nombre de jours facultatif. Sa structure est équivalente à celle du data type Time-of-day, mais elle indique un écart de temps.

Notation Time-difference

Plage de valeurs: $0 \le i \le (2^{28} -1)$ ms
$0 \le i \le (2^{16} -1)$ jours

Codage:

L'heure est représentée par une valeur binaire de 32 bits. Les quatre premiers bits (MSB) doivent avoir la valeur zéro. La date (facultative) est encodée sous la forme d'une valeur binaire de 16 bits (2 octets). Time-difference est une chaîne de 4 ou 6 octets, comme indiqué à la Figure 14.

bits	8	7	6	5	4	3	2	1	
octets 1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	nombre de
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	millisecondes
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	nombre de jours
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	facultatif
	msb								

Figure 14 – Codage des données de type Time-difference

5.1.3.2.11 Codage du type Bit String

La Figure 15 représente le modèle de numérotation des bits du data type Bit String.

Seuls les multiples de huit sont considérés comme des valeurs de longueur (en bits) conformes pour Bit String.

Notation: Bit String
Codage: Binaire

bits	8	7	6	5	4	3	2	1
octets								
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
.	etc.							
n								

Figure 15 – Codage des données de type Bit String

5.1.3.2.12 Codage du type Time-value

Ce type de données permet de représenter la date et l'heure avec la précision exigée pour la synchronisation temporelle des applications. Il correspond à un nombre en virgule fixe sans signe de 64 bits, qui représente l'heure en 1/32 de milliseconde. Lorsque le bit SN est défini sur 1, les données sont négatives et utilisent la représentation par complément à deux; elles sont positives lorsque le bit SN est défini sur 0. Voir Figure 16.

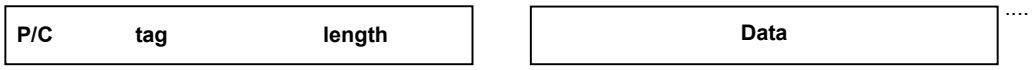
bits	8	7	6	5	4	3	2	1	
octets 1	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}	
2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}	
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}	
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}	
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	entier algébrique de longueur 8 octets, en 1/32 de ms
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	msb								

Figure 16 – Codage des données de type Time-value

5.1.3.3 Définitions de données d'utilisateur

5.1.3.3.1 Généralités

Les données d'utilisateur constituent toujours un composant simple (primitive). Elles sont encodées comme indiqué à la Figure 16. L'indicateur P/C, le marqueur et la longueur sont encodés comme indiqué à la Figure 17.

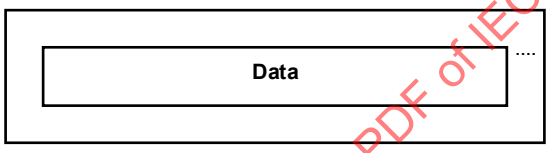


Légende

Anglais	Français
P/C	P/C
tag	repère
length	longueur
Data	Données

Figure 17 – Codage des données de définitions de données d'utilisateur avec identificateur

Si la sémantique des données d'utilisateur est déterminée de manière implicite à partir de la position au sein de l'unité PDU et que la longueur est fixe et déterminée de manière implicite, aucune information d'identification n'est ajoutée (voir Figure 18).



Légende

Anglais	Français
Data	Données

Figure 18 – Codage des données de définitions de données d'utilisateur sans identificateur

Deux définitions spéciales sont utilisées pour décrire les données d'utilisateur dans les définitions de PDUBody: Null et Packed. Elles sont définies ci-dessous.

5.1.3.3.2 Codage de Null

Null possède une longueur nulle. Il n'est suivi d'aucun octet (vide).

5.1.3.3.3 Codage de Packed

Packed contient un ou plusieurs éléments de données issus des data types, qui s'enchaînent sans espace. Sa composition est connue de l'utilisateur.

5.1.3.4 Codage des informations de structure

5.1.3.4.1 Généralités

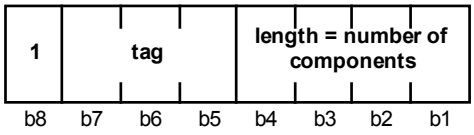
Les données d'utilisateur peuvent être associées à des composants structurés (construits).

Le partenaire de communication doit être capable d'identifier ces structures, ainsi que leurs composants. L'indicateur P/C de l'ID Info est 1.

5.1.3.4.2 SEQUENCE

La structure SEQUENCE est comparable à un enregistrement. Elle représente un ensemble de données d'utilisateur de même type ou de type différent. Une structure ID Info précède la

structure SEQUENCE; elle exprime la longueur non pas en octets, mais en nombre de composants. Le nombre de composants est inférieur à la longueur totale en octets. Dans la plupart des cas, du fait de cet encodage de la longueur, aucune extension n'est nécessaire. Il convient d'encoder les composants selon l'ordre d'ANS.1. Voir Figure 19. Ni la duplication, ni la suppression d'éléments ne sont admises.



Légende

Anglais	Français
tag	repère
length = number of components	longueur = nombre de composants

Figure 19 – Codage de l'ID Info d'une structure SEQUENCE

Une structure peut contenir des données d'utilisateur ou d'autres structures (définissant des composants). Les composants simples peuvent être OPTIONAL, c'est-à-dire qu'ils peuvent être négligés. Dans ce cas, l'ID Info est également négligée. Une structure SEQUENCE doit être comptabilisée comme un composant simple, même si elle contient plusieurs composants.

Exemple:

L'exemple d'encodage ci-dessous utilise la notation hexadécimale. Le X majuscule sert de caractère de remplissage pour les valeurs inconnues telles que la longueur des composants simples ou le marqueur de la structure. Un x minuscule représente des données d'utilisateur.

```
Syntax Description     Code   comment
Person [1] IMPLICIT SEQUENCE      {94     4 components
   [0] Surname,                     0X xx ...   tag 0
   [1] First name,                  1X xx ... tag 1
   [2] City,                        2X xx ... tag 2
   [3] Street                       3X xx ... tag 3
}
```

5.1.3.4.3 SEQUENCE OF

La structure SEQUENCE OF représente une succession de composants. Elle est comparable à un tableau.

La structure peut contenir un ou plusieurs composants. Les composants peuvent être des données d'utilisateur ou des structures.

Le codage de cette structure est similaire à celui de la structure SEQUENCE. Dans l'énoncé du nombre de composants, le nombre de répétitions doit être pris en compte. Les marqueurs de Syntax Description doivent être utilisés pour les composants de la structure SEQUENCE OF. Les mêmes marqueurs peuvent être codés plusieurs fois à la suite.

Exemple:

```
employeedata [2] IMPLICIT SEQUENCE OF {
   [0] Person
}
```

5.1.3.4.4 CHOICE

Un CHOICE (CHOIX) représente une sélection que l'utilisateur effectue parmi un ensemble de possibilités prédéfinies. Les composants d'une construction CHOICE doivent posséder des marqueurs différents afin de permettre une identification correcte. Le composant réellement choisi est encodé à la place de la construction CHOICE. Pour une structure CHOICE, il

convient d'encoder un seul composant. Son marqueur possède une valeur spécifiée dans ASN.1.

Exemple:

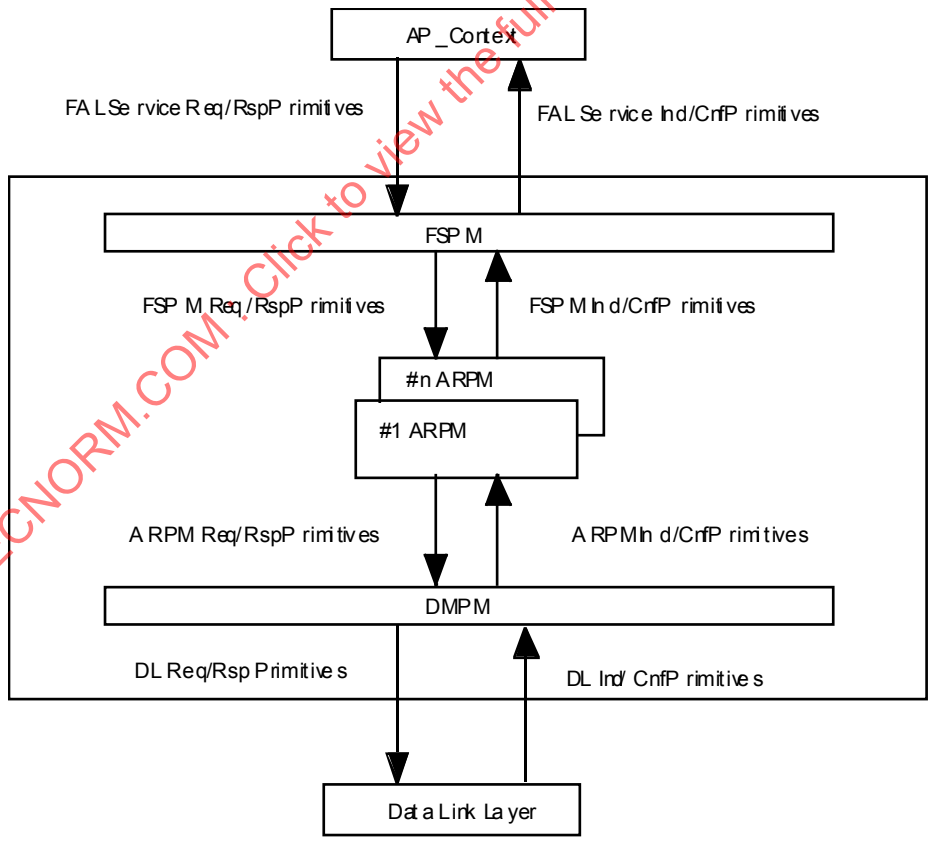
```
Data::= CHOICE {
   [0] Employeedata,
   [1] Clientdata,
   [2] Supplierdata
}
```

6 Structure des diagrammes d'états de protocole de la couche FAL

Le présent article décrit l'interface avec les services et les machines de protocole de couche FAL.

NOTE Les diagrammes d'états spécifiés dans l'Article 6 et les machines ARPM définies dans l'Article 9 définissent uniquement les événements valides pour chacun. Le traitement de ces événements non valides s'effectue localement.

Le comportement de la couche FAL est décrit par trois machines de protocole intégrées. Des combinaisons spécifiques de ces machines de protocole sont définies pour les différents types de points de fin de relations entre applications (Application Relationship End Point, AREP). 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 Machine, ARPM) et machine de protocole de mapping de couche Liaison de données (data-link layer Mapping Protocol Machine, DMPM). La Figure 20 illustre les relations et les primitives échangées entre ces machines de protocole.



Légende

Anglais	Français
AP_Context	Contexte AP
FAL Service Req/Rsp Primitives	Primitives de Req/Rsp de service FAL

Anglais	Français
FAL Service Ind/Cnf Primitives	Primitives d'Ind/de Cnf de service FAL
FSPM	FSPM
FSPM Req/Rsp Primitives	Primitives de Req/Rsp FSPM
FSPM Ind/Cnf Primitives	Primitives d'Ind/de Cnf FSPM
#n ARPM	ARPM n° n
#1 ARPM	ARPM n° 1
ARPM Req/Rsp Primitives	Primitives de Req/Rsp ARPM
ARPM Ind/Cnf Primitives	Primitives d'Ind/de Cnf ARPM
DMPM	DMPM
DL Req/Rsp Primitives	Primitives de Req/Rsp DL
DL Ind/Cnf Primitives	Primitives d'Ind/de Cnf DL
Data Link Layer	Couche Liaison de données

Figure 20 – Relations entre les machines de protocole et les couches adjacentes

La machine FSPM décrit l'interface de service entre le contexte AP et un point AREP particulier. Elle est commune à toutes les classes AREP et ne présente pas de changements d'état. La machine FSPM est chargée des activités suivantes:

a) accepter les primitives de service provenant de l'utilisateur de service FAL et les convertir en primitives FAL internes;

b) sélectionner un diagramme d'états ARPM approprié en fonction des paramètres d'identificateur AREP fournis par le contexte AP et envoyer les primitives FAL internes à la machine ARPM choisie;

c) accepter les primitives FAL internes provenant de la machine ARPM et les convertir en primitives de service à l'intention du contexte AP;

d) remettre les primitives de service FAL au contexte AP en fonction du paramètre d'identificateur AREP associé aux primitives.

La machine ARPM décrit l'établissement et la libération d'une relation AR, ainsi que l'échange d'unités PDU de couche FAL avec une ou plusieurs machines ARPM distantes. La machine ARPM est chargée des activités suivantes:

e) accepter les primitives FAL internes provenant de la machine FSPM et envoyer d'autres primitives FAL internes soit à la machine FSPM, soit à la machine DMPM, selon le type de primitive et de point de fin AREP;

f) accepter les primitives FAL internes provenant de la machine DMPM et les envoyer à la machine FSPM sous la forme de primitives FAL internes;

g) si les primitives concernent le service Establish ou Abort, elle doit essayer d'établir ou de libérer la relation AR indiquée.

La machine DMPM décrit le mapping entre la couche FAL et la couche DLL. Elle est commune à tous les types AREP et ne présente pas de changements d'état. La machine DMPM est chargée des activités suivantes:

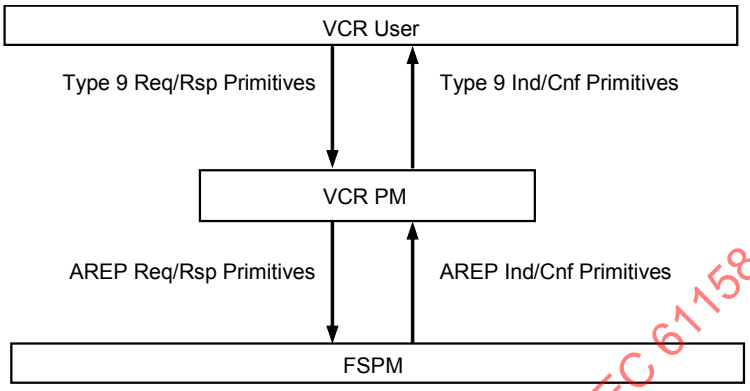
h) accepter les primitives FAL internes provenant de la machine ARPM, préparer les primitives de service DLL et les envoyer à la couche DLL;

i) recevoir les primitives d'indication ou de confirmation provenant de la couche DLL et les envoyer à la machine ARPM sous la forme de primitives FAL internes.

7 Diagrammes d'états de contexte AP

7.1 Structure PM VCR

Ce bus de terrain définit un diagramme d'états unique: la machine de protocole de connexion de relation de communication virtuelle (Virtual Communication Relation, VCR). La relation de cette machine avec son utilisateur et avec la machine FSPM sous-jacente, en indiquant les primitives échangées, illustre la Figure 21.



Légende

Anglais	Français
VCR User	Utilisateur VCR
Type 9 Req/Rsp Primitives	Primitives de Req/Rsp de Type 9
Type 9 Ind/Cnf Primitives	Primitives d'Ind/de Cnf de Type 9
VCR PM	VCR PM
AREP Req/Rsp Primitives	Primitives de Req/Rsp d'AREP
AREP Ind/Cnf Primitives	Primitives d'Ind/de Cnf d'AREP
FSPM	FSPM

Figure 21 – Relations entre les machines de protocole et les couches adjacentes

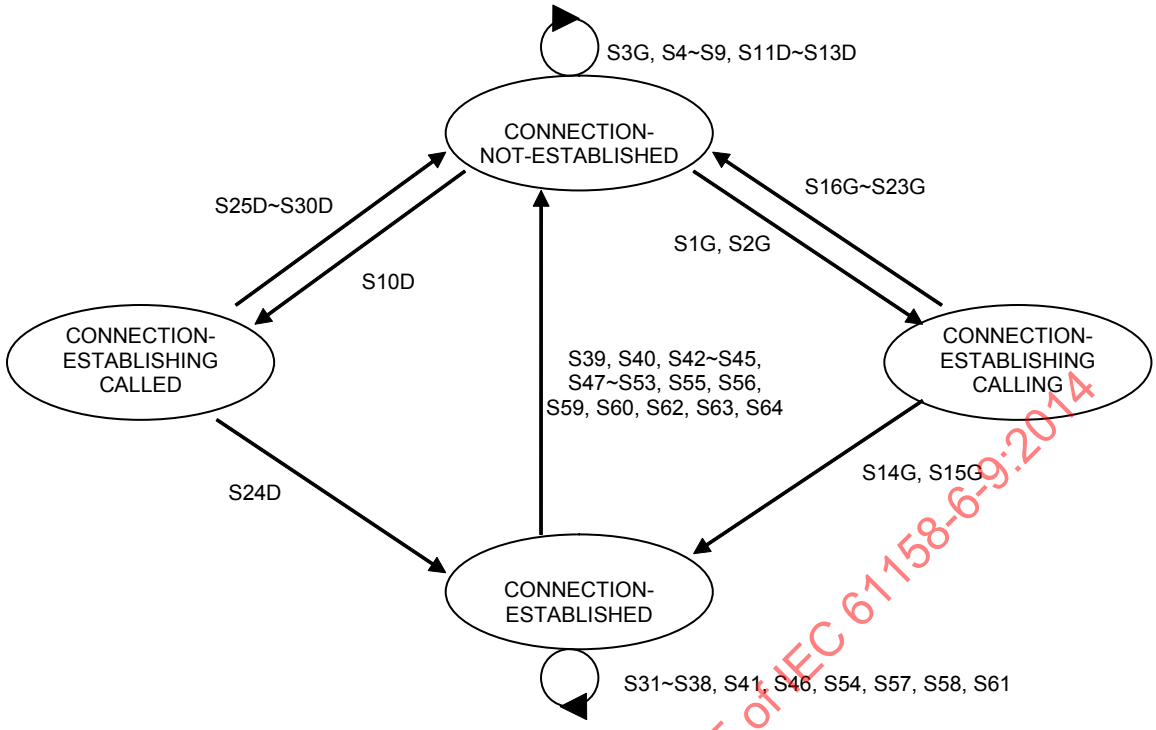
La machine de protocole VCR (PM VCR) est un diagramme d'états PM VCR qui décrit le fonctionnement des VCR établies de manière dynamique par l'échange d'unités APDU de service de lancement. La machine PM VCR est chargée des activités suivantes:

1. accepter les primitives de demande et de réponse de service provenant de l'utilisateur de service VCR;
2. accepter les primitives d'indication et de confirmation de service provenant des machines ARPM;
3. remettre les primitives d'indication et de confirmation de service à l'utilisateur VCR.
4. envoyer les primitives de demande et de réponse de service à la machine ARPM en fonction des règles de ce diagramme d'états.

7.2 Diagramme d'états de la machine PM VCR

7.2.1 Généralités

Le diagramme d'états de la machine PM VCR, représenté à la Figure 22, décrit le comportement du point de fin de relation VCR. Dans le cas des relations VCR sans connexion, le diagramme d'états commence à l'état OUVERT.



Légende

Anglais	Français
CONNECTION-NOT-ESTABLISHED	CONNEXION-NON-ETABLIE
CONNECTION-ESTABLISHINGCALLED	ETABLISSEMENT-CONNEXION (APPELE)
CONNECTION-ESTABLISHING CALLING	ETABLISSEMENT-CONNEXION (APPELANT)
CONNECTION-ESTABLISHED	CONNEXION-ETABLIE

Figure 22 – Diagramme d'états VCR

NOTE Les numéros de transaction portant le suffixe "G" sont destinés à la machine de protocole appelante, ceux portant le suffixe "D", à la machine de protocole appelée. Les transactions dépourvues de suffixe s'appliquent aux deux types de machines de protocole ou aux machines dont le rôle n'a pas encore été déterminé.

7.2.2 Etats des relations VCR AP

CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

La connexion n'est pas établie. Seules les primitives de service Initiate.req, ASC.ind, Abort.req et ABT.ind sont admises. Tous les autres services doivent être rejetés avec le service de coupure.

CONNECTION-ESTABLISHING (CALLING) (ETABLISSEMENT-CONNEXION (APPELANT))

L'utilisateur FMS local souhaite établir la connexion. Seules les primitives de service ASC.cnf(+), ASC.cnf(-), Abort.req et ABT.ind sont admises. Tous les autres services doivent être rejetés avec le service de coupure. Le nom de cet état est abrégé comme suit: ETABLISSEMENT-CON-APPELANT.

CONNECTION-ESTABLISHING (CALLED) (ETABLISSEMENT-CONNEXION (APPELE))

L'utilisateur FMS distant souhaite établir la connexion. Seules les primitives de service Initiate.rsp(+), Initiate.rsp(-), Abort.req et ABT.ind sont admises. Tous les autres services doivent être rejetés avec le service de coupure. Le nom de cet état est abrégé comme suit: ETABLISSEMENT-CON-APPELE.

CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

La relation VCR est établie. Les primitives de service Initiate.req, Initiate.rsp(+) et Initiate.rsp(-) ne sont pas admises; elles doivent être rejetées avec le service de coupure. Pour réinitialiser une relation VCR (Reset VCR), on doit entreprendre les actions suivantes.

- Définir les attributs Outstanding services Counter Sending et Outstanding services Counter Receiving de la VCR (partie dynamique) sur 0.
- Définir l'état de la VCR sur CONNEXION-NON-ETABLIE.

7.2.3 Passages d'état de lancement des relations VCR AP

Le Tableau 3 définit les passages d'état AP-VCR.

Tableau 3 – Transactions du diagramme d'états VCR AP

#	Etat actuel	Evénement ou condition => action	Etat suivant

S1G	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	Initiate.req
		&& VCR is valid
		&& VCR type = QUB
		=>
		ASC.req{
		Data:= Initiate-RequestPDU
		}
			CONNECTION – ESTABLISHING-CALLING

S2G	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	Initiate.req
		&& VCR is valid
		&& VCR type <> QUB
		=>
		ASC.req{
		Data:= NULL
		}
			CONNECTION – ESTABLISHING-CALLING

S3G	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	Initiate.req
		&& VCR is not valid
		=>
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= VCR error
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S4	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	Abort.req => (aucune action entreprise)	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S5	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	FMS service.primitive other than Initiate & Abort
		=>
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= User error
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S6	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	ABT.ind => (aucune action entreprise)	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S7	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FAS error
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S8	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	FAS service primitive other than ASC.ind and ABT.ind
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= Connection state conflict FAS
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant

S9	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	ASC.ind
		&& Data = not allowed, unknown or faulty FMS PDU
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FMS PDU error
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S10D	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	ASC.ind
		&& Data Initiate-RequestPDU
		&& VCR is valid
		&& Max FMS PDU length test is positive
		&& FmsFeaturesSupported test is positive
		=>
		Initiate.ind{
		}
			CONNECTION – ESTABLISHING-CALLED

S11D	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	ASC.ind
		&& Data = Initiate-RequestPDU
		&& VCR is invalid
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= VCR error
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S12D	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	ASC.ind
		&& Data = Initiate-RequestPDU
		&& VCR is valid
		&& Max FMS PDU length test is negative
		=>
		ASC.rsp(-){
		Data:= Initiate-ErrorPDU{
		ErrorCode:= max-fms-pdu-size-insufficient
		}
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S13D	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)	ASC.ind
		&& Data = Initiate-RequestPDU
		&& VCR is valid
		&& Max FMS PDU length test is positive
		&& FmsFeaturesSupported test is negative
		=>
		ASC.rsp(-){
		Data:= Initiate-ErrorPDU{
		ErrorCode:= feature-not-supported
		}
		}
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S14G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(+)
		&& Data = Initiate-ResponsePDU
		&& VCR type = QUB
		=>
		Initiate.cnf(+){
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S15G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(+)
		&& Data = NULL
		&& VCR type = QUU or BNU
		=>
		Initiate.cnf(+){
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S16G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(-) && Data = Initiate-ErrorPDU && VCR type = QUB => Initiate.cnf(-){ ErrorCode:= Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)
S17G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(-) && Data = NULL && VCR type = QUU or BNU => Initiate.cnf(-){ ErrorCode:= Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant
S18G	CONNECTION - ESTABLISHING -CALLING	ABT.ind => Abort.ind{ AbortIdentifier:= AbortIdentifier of ABT.ind, ReasonCode:= ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)
S19G	CONNECTION - ESTABLISHING -CALLING	Abort.req => ABT.req{ AbortIdentifier:= AbortIdentifier of Abort.req, ReasonCode:= ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S20G	CONNECTION - ESTABLISHING -CALLING	Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FAS error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= FAS error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S21G	CONNECTION - ESTABLISHING -CALLING	FAS service primitive other than ASC.cnf and ABT.ind
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:=Connection state conflict FAS
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:=Connection state conflict FAS
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S22G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf
		&& Data = not allowed, unknown or faulty FMS PDU
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FMS PDU error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= FMS PDU error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S23G	CONNECTION - ESTABLISHING -CALLING	FMS service.primitive other than Initiate.rsp and Abort.req
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= User error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= User error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S24D	CONNECTION - ESTABLISHING -CALLED	Initiate.rsp(+)
		=>
		ASC.rsp(+){
		Data = Initiate-ResponsePDU
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S25D	CONNECTION - ESTABLISHING -CALLED	Initiate.rsp(-)
		=>
		ASC.rsp(-){
		Data = Initiate-ErrorPDU{
		ErrorCode = ErrorCode of Initiate.rsp
		}
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant
S26D	CONNECTION - ESTABLISHING -CALLED	Abort.req => ABT.req{ AbortIdentifier:= AbortIdentifier of Abort.req, ReasonCode:= ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)
S27D	CONNECTION - ESTABLISHING -CALLED	ABT.ind => Abort.ind{ AbortIdentifier:= AbortIdentifier of ABT.ind, ReasonCode:= ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S28D	CONNECTION - ESTABLISHING -CALLED	Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FAS error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= FAS error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S29D	CONNECTION - ESTABLISHING -CALLED	FAS service primitive other ABT.ind
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:=Connection state conflict FAS
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:=Connection state conflict FAS
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S30D	CONNECTION - ESTABLISHING -CALLED	FMS service.primitive other than Initiate.rsp and Abort.req
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= User error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= User error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S31	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) positive => DTC.req{ Data:= Confirmed-RequestPDU }, OSCS:= OSCS + 1	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S32	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.req
		&& OSCS = ActualMaxSCC
		=>
		Reject.ind{
		DetectedHere:= true,
		Original InvokeID:= InvokeID in ConfirmedService.req,
		RejectPDUType:= Confirmed-RequestPDU,
		RejectCode:= Max-services-overflow
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant

S33	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.req
		&& OSCS < ActualMaxSCC
		&& InvokeID already existent
		=>
		Reject.ind{
		DetectedHere:= true,
		Original InvokeID:= InvokeID in ConfirmedService.req,
		RejectPDUType:= Confirmed-RequestPDU,
		RejectCode:= Invoke-ID-exists
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S34	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.req
		&& OSCS < ActualMaxSCC
		&& InvokeID not existent
		&& PDU length > Max FMS PDU sending
		=>
		Reject.ind{
		DetectedHere:= true,
		Original InvokeID:= InvokeID in ConfirmedService.req,
		RejectPDUType:= Confirmed-RequestPDU,
		RejectCode:= PDU-size
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S35	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.req
		&& OSCS < ActualMaxSCC
		&& InvokeID not existent
		&& PDU length ≤ Max FMS PDU sending
		&& FmsFeaturesSupported test (client) negative
		=>
		Reject.ind{
		DetectedHere:= true,
		Original InvokeID:= InvokeID in ConfirmedService.req,
		RejectPDUType:= Confirmed-RequestPDU,
		RejectCode:= Feature-not-supported-connection-oriented
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S36	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	UnconfirmedService.req
		&& PDU length ≤ Max FMS PDU sending
		&& FmsFeaturesSupported test (client) positive
		=>
		DTU.req{
		Data:= Unconfirmed-PDU
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S37	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	UnconfirmedService.req
		&& PDU length > Max FMS PDU sending
		=>
		Reject.ind{
		DetectedHere:= true,
		Original InvokeID:= InvokeID in UnconfirmedService.req,
		RejectPDUType:= Unconfirmed-PDU,
		RejectCode:= PDU-size
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S38	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	UnconfirmedService.req
		&& PDU length ≤ Max FMS PDU sending
		&& FmsFeaturesSupported test (client) negative
		=>
		Reject.ind{
		DetectedHere:= true,
		Original InvokeID:= InvokeID in UnconfirmedService.req,
		RejectPDUType:= Unconfirmed-PDU,
		RejectCode:= Feature-not-supported-connection-oriented
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S39	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	Abort.req => ABT.req{ AbortIdentifier:= AbortIdentifier of Abort.req, ReasonCode:= ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant

S40	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	Faulty, unknown or not-allowed FMS service.primitive
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= User error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= User error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S41	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR < ActualMaxRCC
		&& InvokeID not existent
		&& Features Supported test (server) positive
		=>
		ConfirmedService.ind{
		}
		OSCR:= OSCR + 1
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S42	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length > Max FMS PDU receiving
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= PDU-size
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= PDU-size
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S43	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR ≥ ActualMaxRCC
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= Max-services-overflow
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= Max-services-overflow
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S44	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR < ActualMaxRCC
		&& InvokeID already existent
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-request
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-request
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant

S45	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.ind
		&& Data = Confirmed-ServicePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& OSCR < ActualMaxRCC
		&& InvokeID not existent
		&& Features Supported test (server) negative
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= Feature-not-supported
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= Feature-not-supported
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S46	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTU.ind
		&& Data = Unconfirmed-PDU
		&& PDU length ≤ Max FMS PDU receiving
		&& Features Supported test (server) positive
		=>
		UnconfirmedService.ind{
		}
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S47	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTU.ind
		&& Data = Unconfirmed-PDU
		&& PDU length > Max FMS PDU receiving
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= PDU-size
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= PDU-size
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S48	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTU.ind
		&& Data = Unconfirmed-PDU
		&& PDU length ≤ Max FMS PDU receiving
		&& Features Supported test (server) negative
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= Feature-not-supported
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= Feature-not-supported
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S49	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ABT.ind => Abort.ind{ AbortIdentifier:= AbortIdentifier of ABT.ind, ReasonCode:= ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S50	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ASC.ind
		&& Data = Initiate-RequestPDU
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= Connection-state-conflict-FMS
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= Connection-state-conflict-FMS
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant

S51	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	Faulty or unknown FAS service primitive
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FAS error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= FAS error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S52	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	Not-allowed FAS service primitive
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= Connection-state-conflict-FAS
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= Connection-state-conflict-FAS
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S53	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	valid FAS service primitive
		&& Data = not-allowed, unknown or faulty FMS PDU
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FMS-PDU-error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= FMS-PDU-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S54	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.rsp && InvokeID (server) existent && services in .rsp and .ind are identical && PDU length ≤ Max FMS PDU sending => DTC.rsp{ Data:= Confirmed-ResponsePDU }, OSCR:= OSCR – 1	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S55	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.rsp
		&& InvokeID (server) not existent
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-response
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-response
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S56	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.rsp
		&& InvokeID (server) existent
		&& services in .rsp and .ind are not identical
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= service-error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= service-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant

S57	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	ConfirmedService.rsp
		&& InvokeID (server) existent
		&& services in .rsp and .ind are identical
		&& PDU length > Max FMS PDU sending
		=>
		DTC.rsp{
		Data:= Reject-PDU{
		Original InvokeID:= InvokeID in ConfirmedService.req,
		RejectCode:= PDU-size
		}
		},
		OSCR:= OSCR – 1
			CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S58	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.cnf && Data = Confirmed-ResponsePDU && PDU length ≤ Max FMS PDU receiving && InvokeID (client) existent && services in .cnf and .req are identical => ConfirmedService.cnf{ }, OSCR:= OSCR – 1	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S59	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.cnf
		&& Data = Confirmed-ResponsePDU
		&& PDU length > Max FMS PDU receiving
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= PDU-size
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= PDU-size
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S60	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.cnf
		&& Data = Confirmed-ResponsePDU
		&& PDU length ≤ Max FMS PDU receiving
		&& InvokeID (client) not existent
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-response
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-response
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S61	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.cnf && Data = Reject-PDU && Original InvokeID (client) existent && RejectCode:= PDU-size => Reject.ind{ DetectedHere:= false, Original InvokeID:= InvokeID in Reject-PDU, RejectPDUType:= Confirmed-Response-PDU, RejectCode:= PDU-size }, OSCS:= OSCS -1	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)

S62	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.cnf
		&& Data = Reject-PDU
		&& Original InvokeID (client) not existent
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-response
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= InvokeID-error-response
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

#	Etat actuel	Evénement ou condition => action	Etat suivant

S63	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.cnf
		&& Data = Reject-PDU
		&& Original InvokeID (client) existent
		&& RejectCode:<> PDU-size
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= FMS-PDU-error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= FMS-PDU-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

S64	CONNECTION-ESTABLISHED (CONNEXION-ETABLIE)	DTC.cnf
		&& Data = Confirmed-ResponsePDU
		&& InvokeID (client) existent
		&& Services in .req and .cnf are not identical
		=>
		ABT.req{
		AbortIdentifier:= FMS,
		ReasonCode:= Service-error
		},
		Abort.ind{
		AbortIdentifier:= FMS,
		ReasonCode:= Service-error
		},
		reset VCR
			CONNECTION-NOT-ESTABLISHED (CONNEXION-NON-ETABLIE)

7.2.4 Primitives échangées entre l'utilisateur FAL et la machine PM VCR

Le Tableau 4 et le Tableau 5 décrivent les primitives échangées entre l'utilisateur FAL et la machine PM VCR.

Tableau 4 – Primitives adressées par l'utilisateur FAL à la machine PM VCR

Nom de la primitive	Source	Paramètres et fonctions associés
Terminate.req	Utilisateur FAL	Fait référence à la définition de service (voir la CEI 61158-5-9)
Initiate.req	Utilisateur FAL	
Initiate.rsp(+)	Utilisateur FAL	
Initiate.rsp(-)	Utilisateur FAL	
ConfirmedService.req	Utilisateur FAL	
ConfirmedService.rsp	Utilisateur FAL	
UnconfirmedService.req	Utilisateur FAL	

Tableau 5 – Primitives adressées par la machine PM VCR à l'utilisateur FAL

Nom de la primitive	Source	Paramètres et fonctions associés
Terminate.ind	PM VCR	Fait référence à la définition de service (CEI 61158-5-9)
Initiate.ind	PM VCR	
Initiate.cnf(+)	PM VCR	
Initiate.cnf(-)	PM VCR	
ConfirmedService.ind	PM VCR	
ConfirmedService.cnf	PM VCR	
UnconfirmedService.ind	PM VCR	

7.2.5 Primitives échangées entre la machine FSPM et la machine PM VCR

Le Tableau 6 et le Tableau 7 définissent les primitives utilisées par la machine FSPM.

Tableau 6 – Primitives adressées par la machine PM VCR à la machine FSPM

Nom de la primitive	Source	Paramètres associés	Fonctions
ASC.req	PM VCR	Arep_Id, Data, Remote_DLCEP_Address	Cette primitive permet à la machine PM VCR de transmettre une primitive de demande d'association à la machine FSPM.
ASC.rsp(+)	PM VCR	Arep_Id, Data	Cette primitive permet à la machine PM VCR de transmettre une primitive de réponse d'association positive à la machine FSPM.
ASC.rsp(-)	PM VCR	Arep_Id, Data	Cette primitive permet à la machine PM VCR de transmettre une primitive de réponse d'association négative à la machine FSPM.
Abort.req	PM VCR	Arep_Id, Identifier, Reason_Code, Additional_Detail	Cette primitive permet à la machine PM VCR de transmettre une primitive de demande de coupure à la machine FSPM.
CS.req	PM VCR	Arep_Id, Data	Cette primitive permet à la machine PM VCR de transmettre une primitive de demande d'envoi confirmé (Confirmed Send, CS) à la machine FSPM.
CS.rsp	PM VCR	Arep_Id, Data	Cette primitive permet à la machine PM VCR de transmettre une primitive de réponse d'envoi confirmé à la machine FSPM.
UCS.req	PM VCR	Arep_Id, Remote_DLSAP_Address, Data	Cette primitive permet à la machine PM VCR de transmettre une primitive de demande d'envoi non confirmé (UnConfirmed Send, UCS) à la machine FSPM.
FCMP.req	PM VCR	Arep_Id	Cette primitive permet à la machine PM VCR de transmettre une primitive de demande FCMP (FAL-Compel) à la machine FSPM.
GBM.req	PM VCR	Arep_Id	Cette primitive permet à la machine PM VCR de transmettre une primitive de demande GBM (Get-Buffered-Message) à la machine FSPM.

Tableau 7 – Primitives adressées par la machine FSPM à la machine PM VCR

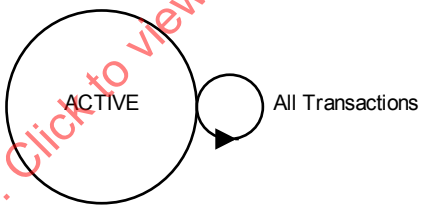
Nom de la primitive	Source	Paramètres associés	Fonctions
ASC.ind	FSPM	Arep_Id, Data	Cette primitive permet à la machine FSPM de transmettre une primitive d'indication d'association à la machine PM VCR.
ASC.cnf(+)	FSPM	Arep_Id, Data	Cette primitive permet à la machine FSPM de transmettre une primitive de résultat d'association positif à la machine PM VCR.
ASC.cnf(-)	FSPM	Arep_Id, Data	Cette primitive permet à la machine FSPM de transmettre une primitive de résultat d'association négatif à la machine PM VCR.
Abort.ind	FSPM	Arep_Id, Locally_Generated, Identifier, Reason_Code, Additional_Detail	Cette primitive permet à la machine FSPM de transmettre une primitive d'indication de coupure à la machine PM VCR.
CS.ind	FSPM	Arep_Id, Data	Cette primitive permet à la machine FSPM de transmettre une primitive d'indication d'envoi confirmé à la machine PM VCR.
CS.cnf	FSPM	Arep_Id, Data	Cette primitive permet à la machine FSPM de transmettre une primitive de confirmation d'envoi confirmé à la machine PM VCR.

Nom de la primitive	Source	Paramètres associés	Fonctions
UCS.ind	FSPM	Arep_Id, Remote_DLSAP_Address, Duplicate_FAL-SDU, Data, Local_Timeliness, Remote_Timeliness	Cette primitive permet à la machine FSPM de transmettre une primitive d'indication d'envoi non confirmé à la machine PM VCR.
FCMP.cnf	FSPM	Arep_Id, Status	Cette primitive permet à la machine FSPM de transmettre une primitive de confirmation FCMP à la machine PM VCR.
GBM.cnf(+)	FSPM	Arep_Id, Duplicate_FAL-SDU, Data, Local_Timeliness, Remote_Timeliness	Cette primitive permet à la machine FSPM de transmettre une primitive de confirmation positive GBM à la machine PM VCR.
GBM.cnf(-)	FSPM	Arep_Id	Cette primitive permet à la machine FSPM de transmettre une primitive de confirmation négative GBM à la machine PM VCR.
FSTS.ind	FSPM	Arep_Id, Reported_Status	Cette primitive permet à la machine FSPM de transmettre une primitive d'indication FSTS (FAL-Status) à la machine PM VCR.

8 Machine de protocole de service FAL (FSPM)

8.1 Généralités

La machine de protocole de service FAS (Fieldbus Access Sublayer) est commune à tous les types de points AREP. Seules les primitives applicables diffèrent d'un type de point AREP à l'autre. La machine possède un seul état appelé "ACTIF" (voir Figure 23).



Légende

Anglais	Français
All Transactions	Toutes les transactions

Figure 23 – Schéma de transition d'états de FSPM

8.2 Tableaux d'états de la machine FSPM

Le Tableau 8 et le Tableau 9 spécifient la machine de protocole FSPM.

Tableau 8 – Tableau d'états de la machine FSPM: transactions expéditeur

#	Etat actuel	Evénement ou condition => action	Etat suivant

S1	ACTIVE	ASC.req
		&& SelectArep (Arep_Id) = "True"
		=>
		ASC_req {
		user_data:= Data,
		remote_dlcep_address:= Remote_DLCEP_Address
		}
			ACTIVE

S2	ACTIVE	ASC.rsp(+)
		&& SelectArep (Arep_Id) = "True"
		=>
		ASC_rsp(+) {
		user_data:= Data
		}
			ACTIVE

S3	ACTIVE	ASC.rsp(-)
		&& SelectArep (Arep_Id) = "True"
		=>
		ASC_rsp(-) {
		user_data:= Data
		}
			ACTIVE

S4	ACTIVE	DTU.req
		&& SelectArep (Arep_Id) = "True"
		=>
		DTU_req {
		remote_dlsap_address:= Remote_DLSAP_Address,
		user_data:= Data
		}
			ACTIVE

S5	ACTIVE	DTC.req
		&& SelectArep (Arep_Id) = "True"
		=>
		DTC_req {
		user_data:= Data
		}
			ACTIVE

S6	ACTIVE	DTC.rsp
		&& SelectArep (Arep_Id) = "True"
		=>
		DTC_rsp {
		user_data:= Data
		}
			ACTIVE

S7	ACTIVE	Abort.req
		&& SelectArep (Arep_Id) = "True"
		=>
		Abort_req {
		identifier:= Identifier,
		reason_code:= Reason_Code,
		additional_detail:= Additional_Detail
		}
			ACTIVE

S8	ACTIVE	FCMP.req
		&& SelectArep (Arep_Id) = “True”
		=>
		FCMP_req {
		}
			ACTIVE

S9	ACTIVE	GBM.req
		&& SelectArep (Arep_Id) = “True”
		=>
		GBM_req {
		}
			ACTIVE

NOTE 1 Une primitive générée dans le diagramme d'états expéditeur de la machine FSPM est envoyée à une machine ARPM appropriée, choisie par la machine FSPM au moyen de la fonction SelectArep. Le paramètre Arep_Id fourni par l'utilisateur FAS constitue l'argument de cette fonction.

NOTE 2 Si la fonction SelectArep renvoie la valeur False, la transmission de cette instance s'effectue localement et la machine FSPM ne génère aucune primitive à l'intention de la machine ARPM.

Tableau 9 – Tableau d'états de la machine FSPM: transactions destinataire

#	Etat actuel	Evénement ou condition => action	Etat suivant

R1	ACTIVE	ASC_ind
		=>
		ASC.ind {
		Arep_Id:= arep_id,
		Data:= user_data
		}
			ACTIVE

R2	ACTIVE	ASC_cnf(+)
		=>
		ASC.cnf(+) {
		Arep_Id:= arep_id,
		Data:= user_data
		}
			ACTIVE

R3	ACTIVE	ASC_cnf(-)
		=>
		ASC.cnf(-) {
		Arep_Id:= arep_id,
		Data:= user_data
		}
			ACTIVE

R4	ACTIVE	DTU_ind
		=>
		DTU.ind {
		Arep_Id:= arep_id,
		Remote_DLSAP_Address:= remote_dlsap_address,
		Duplicate_FAS-SDU:= duplicate_fas_sdu,
		Data:= user_data,
		Local_Timeliness:= local_timeliness,
		Remote_Timeliness:= remote_timeliness
		}
			ACTIVE

R5	ACTIVE	DTC_ind
		=>
		DTC.ind {
		Arep_Id:= arep_id,
		Data:= user_data
		}
			ACTIVE

R6	ACTIVE	DTC_cnf
		=>
		DTC.cnf {
		Arep_Id:= arep_id,
		Data:= user_data
		}
			ACTIVE

R7	ACTIVE	Abort_ind
		=>
		Abort.ind {
		Arep_Id:= arep_id,
		Locally_Generated:= locally_generated,
		Identifier:= identifier,
		Reason_Code:= reason_code,
		Additional_Detail:= additional_detail
		}
			ACTIVE

R8	ACTIVE	FCMP_cnf
		=>
		FCMP.cnf {
		Arep_Id:= arep_id,
		Status:= status
		}
			ACTIVE

R9	ACTIVE	GBM_cnf(+)
		=>
		GBM.cnf(+) {
		Arep_Id:= arep_id,
		Duplicate_FAS-SDU:= duplicate_fas_sdu,
		Data:= user_data,
		Local_Timeliness:= local_timeliness,
		Remote_Timeliness:= remote_timeliness
		}
			ACTIVE

R10	ACTIVE	GBM_cnf(-)
		=>
		GBM.cnf(-) {
		Arep_Id:= arep_id,
		}
			ACTIVE

#	Etat actuel	Evénement ou condition => action	Etat suivant

R11	ACTIVE	FSTS_ind
		=>
		FSTS.ind {
		Arep_Id:= arep_id,
		Reported_Status:= reported_status
		}
			ACTIVE

8.3 Fonctions utilisées par la machine FSPM

Le Tableau 10 définit la fonction utilisée par la machine FSPM.

Tableau 10 – Fonction SelectArep()

Nom	SelectArep	Utilisée dans	FSPM
Entrée		Sortie	
Arep_Id		True || False	
Fonction			
Recherche l'article AREP spécifié par le paramètre Arep_Id. Le paramètre Arep_Id est fourni avec les primitives de service d'utilisateur FAS.			

8.4 Paramètres des primitives FSPM/ARPM

Les paramètres utilisés avec les primitives échangées entre la machine FSPM et la machine ARPM sont décrits dans le Tableau 11.

Tableau 11 – Paramètres utilisés avec les primitives échangées entre la machine FSPM et la machine ARPM

Nom de paramètre	Description
arep_id	Ce paramètre permet d'identifier de manière absolue une instance du point AREP qui a envoyé une primitive. La présente spécification ne spécifie aucune méthode permettant cette identification.
user_data	Ce paramètre transmet les données d'utilisateur FAS.
locally_generated	Ce paramètre transmet la valeur utilisée pour le paramètre Locally_Generated.
identificateur	Ce paramètre transmet la valeur utilisée pour le paramètre Identifier.
reason_code	Ce paramètre transmet la valeur utilisée pour le paramètre Reason_Code.
additional_detail	Ce paramètre transmet la valeur utilisée pour le paramètre Additional_Detail.
duplicate_fas_sdu	Ce paramètre transmet la valeur utilisée pour le paramètre Duplicate_FAS-SDU.
remote_dlsap_address	Ce paramètre transmet la valeur utilisée pour le paramètre Remote_DLSAP_Address.
état	Ce paramètre transmet la valeur utilisée pour le paramètre Status.
reported_status	Ce paramètre transmet un état d'événement Data Like Layer.
local_timeliness	Ce paramètre transmet la valeur utilisée pour le paramètre Local_Timeliness.
remote_timeliness	Ce paramètre transmet la valeur utilisée pour le paramètre Remote_Timeliness.

9 Machines de protocole de relations entre applications (ARPM)

9.1 Mapping de point AREP avec la couche Liaison de données

9.1.1 Généralités

Le Paragraphe 9.1 décrit le mapping de la sous-couche FAS avec la couche Liaison de données de bus de terrain. Elle ne redéfinit pas les attributs DLSAP et DLME (Data Link Management Entity) qui sont ou seront définis dans la spécification de couche Liaison de données; elle définit la manière dont chaque classe de relations AR utilise ces attributs. La gestion de réseau, entre autres méthodes, permet de configurer et de surveiller les valeurs de ces attributs.

Les définitions de classe suivantes décrivent les attributs DLSAP et DLME exigés pour à la prise en charge de chaque classe de points AREP.

NOTE Les attributs non définis utilisent les mêmes définitions que les attributs déjà définis.

9.1.2 Mapping DLL de la classe de points AREP QUU

9.1.2.1 Modèle formel de la classe de points AREP QUU

Le présent paragraphe définit les attributs de mapping DLL, les valeurs admises correspondantes et les services DLL utilisés avec la classe de points AREP QUU.

CLASSE: **QuuCl**
CLASSE PARENTE: QueuedUser-TriggeredUnidirectionalAREP
ATTRIBUTS:

1.	(m)	KeyAttribute:	LocalDlsapAddress
2.	(m)	Attribut:	RemoteDlsapAddress
3.	(m)	Attribut:	LocalDlsapRole (Basic, Group)
4.	(c)	Contrainte:	Role = Source
4.1	(m)	Attribut:	DefaultQosAsSender
4.2	(m)	Attribut	DllPriority (Urgent, Normal, TimeAvailable)
4.3	(m)	Attribut:	MaxConfirmDelayOnUnitdata
4.4	(m)	Attribut:	DlpduAuthentication (Ordinary, Source, Maximal)
4.5	(m)	Attribut:	DlSchedulingPolicy (Implicit)
4.6	(m)	Attribut:	ExplicitQueue (True, False)
5.	(c)	Contrainte:	ExplicitQueue = True
5.1	(m)	Attribut:	QueueBindings—pour un expéditeur ou un destinataire
5.2	(m)	Attribut:	QueueIdentifier
5.3	(m)	Attribut:	MaxQueueDepth
5.4	(m)	Attribut:	MaxDlsduSize

SERVICES DLL:

1.	(m)	OpsService:	DL-Unitdata
2.	(c)	Contrainte:	ExplicitQueue = True
3.	(m)	OpsService:	DL-Get

9.1.2.2 Attributs

9.1.2.2.1 LocalDlsapAddress

Cet attribut spécifie l'adresse DLSAP à laquelle ce point AREP est associé. Cet attribut correspond à une adresse DLSAP si l'attribut Role est défini sur Source; il correspond soit à une adresse DL de groupe, soit à une adresse DLSAP si l'attribut Role est défini sur Sink.

Il fournit la valeur du paramètre "DL(SAP)-address" spécifié dans la couche DLL.

Cet attribut contient les trois sous-attributs suivants: Link Address, Node Address et Selector.

9.1.2.2.2 RemoteDlsapAddress

Cet attribut spécifie l'adresse distante à laquelle les unités PDU FAS sont envoyées (AREP Source) ou d'où proviennent les PDU FAS reçues (AREP Sink).

Si l'attribut ConfigurationType est défini sur Linked, la valeur de l'attribut RemoteDlsapAddress a été configurée. S'il est défini sur Free, cette valeur est fournie avec une demande de service.

Cet attribut contient les trois sous-attributs suivants: Link Address, Node Address et Selector.

9.1.2.2.3 LocalDlsapRole

Cet attribut spécifie le comportement du point DLSAP local à utiliser. Si l'attribut Role est défini sur Source, l'attribut LocalDlsapRole prend la valeur Basic. Si l'attribut Role est défini sur Sink, il prend la valeur Basic ou Group.

Il fournit la valeur du paramètre "DL(SAP)-role" spécifié par la couche DLL.

9.1.2.2.4 DefaultQosAsSender

Les attributs DefaultQosAsSender spécifient la qualité de service DLL utilisée par le point AREP émetteur. La couche DLL réceptrice doit prendre en charge la qualité de service spécifiée par ces attributs.

9.1.2.2.5 DllPriority

Cet attribut définit la priorité DLL, limitant ainsi la longueur maximum d'une unité PDU FAS, du chemin de transport d'une relation AR.

Il fournit la valeur du paramètre "DLL priority" de la couche DLL. Les valeurs Urgent, Normal et Time-Available correspondent respectivement aux valeurs URGENT, NORMAL et TIME-AVAILABLE définies dans la CEI 61158-3-1 et la CEI 61158-4-1.

NOTE Il n'est pas possible d'utiliser des priorités différentes pour chacune des unités PDU FAS envoyées par le même point AREP QUU.

9.1.2.2.6 MaxConfirmDelayOnUnitdata

Cet attribut spécifie le délai de confirmation maximum autorisé pour une confirmation locale provenant d'une primitive de demande DL-Unitdata.

Il fournit la valeur du paramètre "Max confirm delay on locally-confirmed DL-Unitdata" de la couche DLL.

9.1.2.2.7 DlpduAuthentication

Cet attribut spécifie la limite inférieure de la longueur des adresses DL qui doivent être utilisées par la couche DLL.

Il fournit la valeur du paramètre "DLPDU authentication" de la couche DLL. Les valeurs Ordinary, Source et Maximal correspondent respectivement aux valeurs ORDINARY, SOURCE et MAXIMAL définies dans la CEI 61158-3-1 et la CEI 61158-4-1.

9.1.2.2.8 DlSchedulingPolicy

Cet attribut guide la couche DLL dans la planification requise par une relation AR. Pour ce point AREP, la couche DLL essaie de transmettre l'unité PDU FAS envoyée par la FAS, dès réception.

Cet attribut fournit la valeur de l'attribut DL-Scheduling-policy. Seule la valeur Implicit est utilisée. Elle correspond à la valeur IMPLICIT définie dans la CEI 61158-3-1 et la CEI 61158-4-1.

9.1.2.2.9 ExplicitQueue

Lorsqu'il porte la valeur True, cet attribut spécifie que les caractéristiques des files d'attente émettrices et réceptrices associées sont configurées et gérées de manière explicite par la gestion de réseau. La valeur False signifie que des files d'attente de profondeur et de longueur propres à la mise en œuvre sont fournies par la couche DLL.

9.1.2.2.10 QueueBindings

Les attributs suivants spécifient la file d'attente explicite rattachée à ce point DLSAP. Pour un expéditeur, la file d'attente est destinée à l'envoi. Pour un destinataire, elle est destinée à la réception.

9.1.2.2.11 QueueIdentifier

Cet attribut fournit une méthode locale permettant d'identifier une file d'attente associée à ce point DLSAP.

Il fournit la valeur du paramètre "Buffer-or-queue-identifier" défini dans la couche DLL.

9.1.2.2.12 MaxQueueDepth

Cet attribut spécifie le nombre maximum d'unités PDU FAS pouvant être mises en file d'attente d'envoi ou de réception.

Il fournit la valeur du paramètre "Maximum queue depth" de la couche DLL.

9.1.2.2.13 MaxDlsduSize

Cet attribut spécifie la longueur maximum d'une unité PDU FAS pouvant être envoyée ou reçue par la couche DLL.

Il fournit la valeur du paramètre "Maximum DLSDU size" de la couche DLL.

9.1.2.3 Services DLL

Pour connaître les descriptions des services DLL, voir la CEI 61158-3-1.

9.1.3 Mapping DLL de la classe de points AREP QUB

9.1.3.1 Modèle formel de la classe de points AREP QUB

Le présent paragraphe définit les attributs de mapping DLL, les valeurs admises correspondantes et les services DLL utilisés avec la classe de points AREP QUB.

CLASSE: QubCo
CLASSE PARENTE: QueuedUser-TriggeredBidirectionalAREP
ATTRIBUTS:

1.	(m)	KeyAttribute:	LocalDlcepAddress
2.	(m)	Attribut:	RemoteDlcepAddress
3.	(m)	KeyAttribute:	DlcepDlIdentifier
4.	(m)	Attribut:	DlsapRole (Basic)
5.	(m)	Attribut:	QosParameterSet
5.1	(m)	Attribut:	DlcepClass (Peer)
5.2	(m)	Attribut:	DlcepDataDeliveryFeatures
5.2.1	(m)	Attribut:	FromRequesterToResponder (Classical, Disordered)
5.2.2	(m)	Attribut:	FromResponderToRequester (Classical, Disordered)
5.3	(m)	Attribut:	Priorité
5.3.1	(m)	Attribut:	DllPriority (Urgent, Normal, TimeAvailable)
5.3.2	(m)	Attribut:	DllPriorityNegotiated (Urgent, Normal, TimeAvailable)
5.4	(m)	Attribut:	DlpduAuthentication (Ordinary, Source, Maximal)
5.5	(m)	Attribut:	ResidualActivity
5.5.1	(m)	Attribut:	ResidualActivityAsSender (True, False)
5.5.2	(m)	Attribut:	ResidualActivityAsReceiver (True, False)
5.6	(m)	Attribut:	MaxConfirmDelay
5.6.1	(m)	Attribut:	MaxConfirmDelayOnDlConnect
5.6.2	(m)	Attribut:	MaxConfirmDelayOnDlData
5.7	(m)	Attribut:	DlSchedulingPolicy (Implicit)
5.8	(m)	Attribut:	ExplicitQueue (True, False)
5,9	(c)	Contrainte:	ExplicitQueue = True
5.9.1	(m)	Attribut:	MaxDlsduSizes
5.9.2	(m)	Attribut:	MaxDlsduSizeFromRequester
5.9.3	(m)	Attribut:	MaxDlsduSizeFromResponder
5.9.4	(m)	Attribut:	MaxDlsduSizeFromRequesterNegotiated
5.9.5	(m)	Attribut:	MaxDlsduSizeFromResponderNegotiated
5.10	(m)	Attribut:	QueueBindings
5.10.1	(m)	Attribut:	SendingBufferOrQueueIdentifier