IEC 61158-6-8

Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications –
Part 6-8: Application layer protocol specification – Type 8 elements

Réseaux de communication industriels – Spécifications de bus de terrain – Partie 6-8: Spécification de protocole de couche application – Éléments de Type 8

IEC 61158-6-8:2007

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

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

About the IEC

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

About IEC publications

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

Useful links:

IEC publications search - www.iec.ch/searchpub

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It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available on-line and also once a month by email.

Electropedia - www.electropedia.org

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

Customer Service Centre - webstore.iec.ch/csc

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

A propos de la CEI

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

A propos des publications CEI

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

Liens utiles:

Recherche de publications CEI - www.iec.ch/searchpub

La recherche avancée vous permet de trouver des publications CEI en utilisant différents critères (numéro de référence, texte, comité d'études,…).
Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

Service Clients - webstore.iec.ch/csc

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

IEC 61158-6-8

Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 6-8: Application layer protocol specification – Type 8 elements

Réseaux de communication industriels – Spécifications de bus de terrain – Partie 6-8: Spécification de protocole de couche application – Éléments de Type 8

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XD
CODE PRIX

ICS 25.040.40; 35.100.70

ISBN 978-2-83220-636-2

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-8: Application layer protocol specification – Type 8 elements

FOREWORD

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

NOTE Use of some of the associated protocol types is restricted by their 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 particular data-link layer protocol type to be used with physical layer and application layer protocols in Type combinations as specified explicitly in the IEC 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

IEC draws attention to the fact that it is claimed that compliance with this standard may involve the use of patents as follows, where the [xx] notation indicates the holder of the patent right:

Type 8:

DE 197 39 297 C2	[PxC]	"Automatisierungssystem und Steuervorrichtung zur transparenten Kommunikation zwischen verschiedenen Netzwerken."
US 2002/0042845 A1	[PxC]	"Automation System and connecting Apparatus for the Transparent Communication between two Networks."

The IEC takes no position concerning the evidence, validity and scope of these patent rights.:

The holders of these patent rights have assured the IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holders of these patent rights are registered with the IEC. Information may be obtained from:

[PxC]: Phoenix Contact GmbH & Co. KG
Intellectual Property Licenses & Standards
Flachsmarktstr. 8
D-32825 Blomberg,
Germany

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61158-6-8 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This first edition and its companion parts of the IEC 61158-6 subseries cancel and replace IEC 61158-6:2003. This edition of this part constitutes a technical revision.

This edition of IEC 61158-6 includes the following significant changes from the previous edition:

a) deletion of the former Type 6 fieldbus for lack of market relevance;

b) addition of new types of fieldbuses;

c) partition of part 6 of the third edition into multiple parts numbered -6-2, -6-3, …

This bilingual version (2013-02) corresponds to the monolingual English version, published in 2007-12.

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

FDIS	Report on voting
65C/476/FDIS	65C/487/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.

The French version of this standard has not been voted upon.

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

The committee has decided that the contents of this publication will remain unchanged until the maintenance result 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.

NOTE The revision of this standard will be synchronized with the other parts of the IEC 61158 series.

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

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/TR 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 6-8: Application layer protocol specification – Type 8 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 8 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 specifies interactions between remote applications and defines the externally visible behavior provided by the Type 8 fieldbus application layer in terms of

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

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

1) define the wire-representation of the service primitives defined in IEC 61158-5-8, and
2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 8 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) 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 61158-5-8.

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 the IEC 61158-6 series.

1.3 Conformance

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

2 Normative references

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

IEC 60559, *Binary floating-point arithmetic for microprocessor systems*

IEC 61158-3-8, *Industrial communication networks – Fieldbus specifications – Part 3-8: Data-link layer service definition – Type 8 elements*

IEC 61158-4-8, *Industrial communication networks – Fieldbus specifications – Part 4-8: Data-link layer protocol specification – Type 8 elements*

IEC 61158-5-8, *Industrial communication networks – Fieldbus specifications – Part 5-8: Application layer service definition – Type 8 elements*

ISO/IEC 7498 (all parts), *Information technology – Open Systems Interconnection – Basic Reference Model*

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

ISO/IEC 8824, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

ISO/IEC 8825, *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*

3 Terms and definitions

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

3.1 ISO/IEC 7498-1 terms

This standard is partly based on the concepts developed in ISO/IEC 7498-1, and makes use of the following terms defined therein:

3.2
application entity

3.3
application process

3.4
application protocol data unit

3.5
application service element

3.6
application entity invocation

3.7
application transaction

3.8
transfer syntax

3.9 ISO/IEC 8822 terms

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

3.10
abstract syntax

3.11 ISO/IEC 9545 terms

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

3.12
application-context

3.13
application-process-type

3.14
application-service-element

3.15
application control service element

3.16 ISO/IEC 8824 terms

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

3.17
any type

3.18
bitstring type

3.19
Boolean type

3.20
choice type

3.21
component type

3.22
false

3.23
integer type

3.24
module

3.25
null type

3.26
object identifier

3.27
octetstring type

3.28
production

3.29
simple type

3.30
sequence of type

3.31
sequence type

3.32
structured type

3.33
tag

3.34
tagged type

3.35
true

3.36
type

3.37 ISO/IEC 8825 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8825 apply:

3.38
encoding (of a data value)

3.39
data value

3.40 Terms and definitions from IEC 61158-5-8

3.41
application relationship

3.42
client

3.43
error class

3.44
publisher

3.45
server

3.46
subscriber

3.47 Other terms and definitions

The following terms and definitions are used in this standard.

3.48
called
service user or a service provider that receives an indication primitive or a request APDU

3.49
calling
service user or a service provider that initiates a request primitive or a request APDU

3.50
management information
network accessible information that supports the management of the Fieldbus environment

3.51
receiving
service user that receives a confirmed primitive or an unconfirmed primitive, or a service provider that receives a confirmed APDU or an unconfirmed APDU

3.52
resource
resource is a processing or information capability of a subsystem

3.53
sending
service user that sends a confirmed primitive or an unconfirmed primitive, or a service provider that sends a confirmed APDU or an unconfirmed APDU

4 FAL syntax description

4.1 FAL-AR PDU abstract syntax

4.1.1 Top level definition

The productions defined here shall be used with the Peripherals Encoding Rules (see 5.2) for APDU encoding.

```
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 Establish service

```
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
}
```

4.1.6 MaxOSCC

```
MaxOSCC ::= Unsigned8
```

4.1.7 MaxOSCS

```
MaxOSCS ::= Unsigned8
```

4.1.8 MaxUCSC

```
MaxUCC ::= Unsigned8
```

4.1.9 MaxUCSS

```
MaxUCS ::= Unsigned8
```

4.1.10 CIU

```
CIU ::= Unsigned32
```

4.1.11 InvokeID

```
InvokeID ::=  Unsigned8
```

4.1.12 ConType

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

4.1.13 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,
}
```

4.1.14 Protocol-Code

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

4.1.15 Block-Number

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

4.1.16 Block-Length

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

4.1.17 Protocol-Data

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

4.1.18 Address2 ARP

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

4.1.19 Destination-Address

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

4.1.20 Source-Address

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

4.1.21 Destination-Node

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

4.1.22 Source-Node

Source-Node ::= Unsigned8

4.1.23 Destination-Subnode

Destination-Subnode ::= Unsigned8

4.1.24 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

```
ErrorType ::= SEQUENCE {
   errorClass                    [0] IMPLICIT ErrorClass,
   optionalParametersMap         [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
   additionalCode                [1] IMPLICIT Integer16 OPTIONAL,
   additionalDescription         [2] IMPLICIT VisibleString OPTIONAL,
}
```

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

```
ErrorClass ::=  CHOICE
{
    vfdState                [1]   IMPLICIT  Integer8 {
                                      other                           (0)
    },
    applicationReference    [2]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      application-unreachable         (1),
                                      application-reference-invalid   (2),
                                      context-unsupported             (3)
    },
    definition              [3]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      object-undefined                (1),
                                      object-attributes-inconsistent  (2),
                                      name-already-exists             (3),
                                      type-unsupported                (4),
                                      type-inconsistent               (5)
    },
    resource                [4]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      memory-unavailable              (1)
    },
    service                 [5]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      object-state-conflict           (1),
                                      pdu-size                        (2),
                                      object-constraint-conflict      (3),
                                      parameter-inconsistent          (4),
                                      illegal-parameter               (5)
    },
    access                  [6]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      object-invalidated              (1),
                                      hardware-fault                  (2),
                                      object-access-denied            (3),
                                      invalid-address                 (4),
                                      object-attribute-inconsistent   (5),
                                      object-access-unsupported       (6),
                                      object-non-existent             (7),
                                      type-conflict                   (8),
                                      named-access-unsupported        (9),
                                      access-to-element-              (10)
                                      unsupported
    },
    objectDescription       [7]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      name-length-overflow            (1),
                                      od-overflow                     (2),
                                      od-write-protected              (3),
                                      extension-length-overflow       (4),
                                      od-description-length-          (5),
                                      overflow
                                      operational-problem             (6)
    },
    conclude                [9]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      further-communication-          (1)
                                      required
    },
    other                   [8]   IMPLICIT  Integer8 {
                                      other                           (0)
    }
}
```

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.


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

```
ReasonCode ::= Unsigned8
```

4.3.2.1.1 Additional detail

```
AdditionalDetaill ::= OctetString
```

4.3.2.2 AREP

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

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 General types

4.3.5.1.1 Gn_Deletable

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

4.3.5.1.2 Gn_Reusable

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

4.3.5.1.3 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
}
```

4.3.5.1.4 Gn_Length

```
Gn_Length ::= Unsigned8
```

4.3.5.1.5 Gn_MoreFollows

```
Gn_MoreFollows ::= Boolean
```

4.3.5.1.6 Gn_Name

```
Gn_Name ::= OctetString
```

4.3.5.1.7 Gn_NumericID

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

4.3.5.1.8 Gn_ObjectDefinition

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

4.3.5.1.9 Gn_SubIndex

```
Gn_SubIndex ::= Unsigned8
```

4.3.5.2 Gn_ObjectClass

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

4.3.5.3 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
}
```

4.4 Object definitions

4.4.1 Top level definition

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

4.4.2 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
}
```

4.4.3 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
   }
}
```

4.4.4 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
}
```

4.4.5 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
}
```

4.5 Abstract syntax of Data types

4.5.1 Notation for the Boolean type

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

4.5.2 Notation for the Integer type

```
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
```

4.5.3 Notation for the Unsigned type

```
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
```

4.5.4 Notation for the Floating Point type

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

4.5.5 Notation for the BitString type

```
BitString ::= BIT STRING           -- For generic use
```

4.5.6 Notation for the OctetString type

```
OctetString ::= OCTET STRING       -- For generic use
```

4.5.7 Notation for VisibleString type

```
VisibleString ::= VISIBLE STRING   -- For generic use
```

4.5.8 Notation for TimeOfDay type

```
TimeOfDay ::= OctetString6
```

4.5.9 Notation for TimeDifference type

```
TimeDifference ::= OctetString6
```

4.5.10 Notation for BinaryDate2000 type

BinaryDate2000 ::= OctetString8

5 Transfer syntax

5.1 Peripherals encoding rules (PER)

The Peripherals Encoding Rule is the encoding of the PDU types defined in 4.1.

5.2 Encoding of APDU types

5.2.1 Overview of APDU encoding

The APDUs encoded with the PER shall have a uniform format. The APDUs shall consist of two major parts, the “APDU Header” part and the “APDU Body” part as shown in Figure 1.

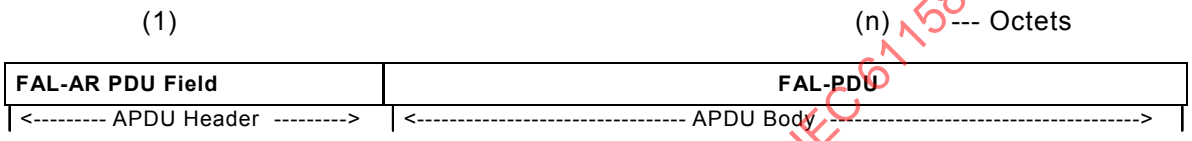


Figure 1 – APDU overview

5.2.2 APDU header encoding

In 4.1, an APDU header is defined as a tagged type of the tag class 'APPLICATION'.

The APDU Header (also called FAL-AR PDU Field) shall be encoded using one or more octets as described in this subclause.

Within the APDU header, the PDU Type tag (e.g. Establish-RequestPDU, ConfirmedSend-RequestPDU...) and the address information (AddressAREP) is encoded .

If the value of the PDU type is < 3, the APDU header shall be encoded using one octet as defined in Figure 2.

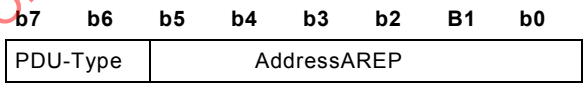


Figure 2 – APDU header

If the value of the PDU type is >= 3 an extension octet is used for the encoding of the tag. The use of a tag extension octet shall be indicated by setting both bits in the PDU type field of the first octet to 1. If the address information (AddressAREP) is greater than 62, an address extension octet shall be used for the encoding of the address value. In this case, the use of an address extension shall be indicated by setting all bits of the address field in the first octet to 1 (see Figure 3 and Figure 4).

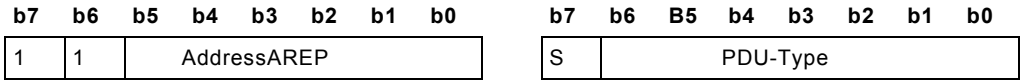


Figure 3 – PDU with type extension

The S field is reserved for system management extensions. For FAL-PDUs this flag is set to 0.

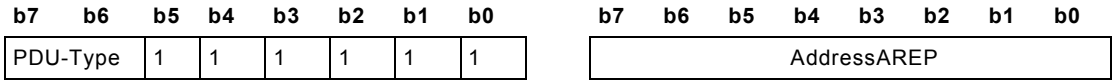


Figure 4 – PDU with address extension

Figure 5 shows the encoding of an APDU header using the tag extension and the address extension octet.

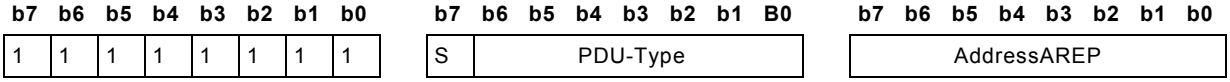


Figure 5 – PDU with type and length extension

Special encoding of the Establish-Request PDU type:

The encoding of the Establish-Request PDU differs from the rules described above, since in this PDU additional parameters, which contain the AR context, are passed. The parameters are the connection type (master-master connection for acyclic data transfer), the parameter CIU (Control Interval User-triggerd) and the outstanding service counters. The connection type parameter is encoded in the most-significant three bits. The least-significant 5 bits contain the number of the additional establish parameters.

If the tag for the address requires an extension (the Establish PDU type has an extension), a minimum PDU length of 9 octets is used in an establish request (see Figure 6).

Example for Establish-RequestPDU:

Establish-RequestPDU tag : 4
AddressAREP : 64
ConnectionType : 0 for MMAZ
CIU : FFFFFFFF
S : 0
MaxOSCC : 1
MaxOSCS : 1
MaxUCSC : 1
MaxUCSS : 1

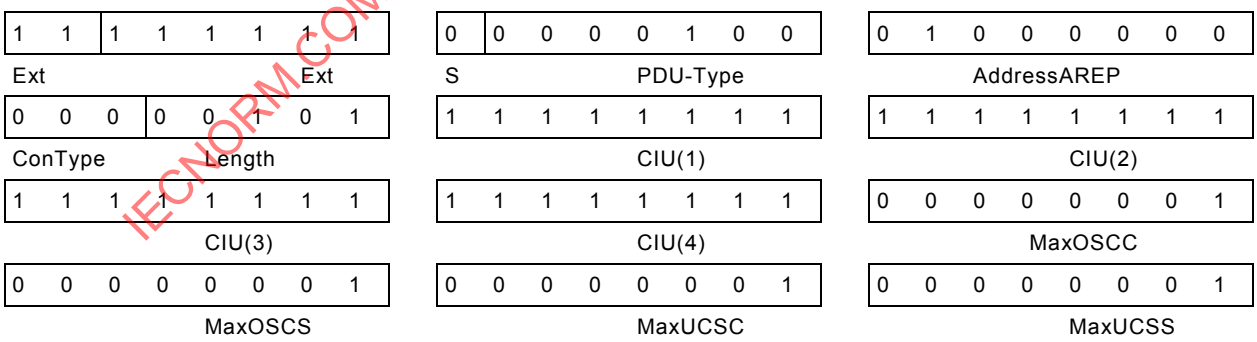


Figure 6 – Example of an Establish-Request PDU

5.2.3 APDU body encoding

The format of the APDU body is described in 4.1 using the ASN.1 syntax. In the following clauses the encoding of the used language elements is described.

5.3 Encoding of tagged type values

5.3.1 Encoding of tagged type values of tag class PRIVATE

Tagged type values with tags of class PRIVATE shall be encoded in the same way as their base types. Tags of class PRIVATE are not explicit encoded (see Figure 7). They are used only to reach syntactical correctness of the ASN.1 definitions.

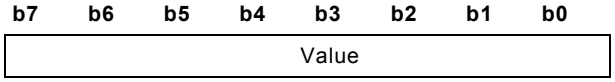


Figure 7 – Encoding of a PRIVATE tagged value

The following restrictions exist for the use of PRIVATE tagged types.

- The length and definition of the encoding of a PRIVATE tagged type shall not change.

- In a SEQUENCE construct no more than one PRIVATE tagged type may be OPTIONAL.

- A private tagged type shall not be contained in a CHOICE construct.

5.3.2 Encoding of context specific tagged type values

5.3.2.1 Overview

Context specific tagged type values are coded as a sequence of one or more Identification information (ID-info) octets, followed by one ore more Value octets (ContentsOctets) as defined in the following clauses. The resulting format is shown in Figure 8.

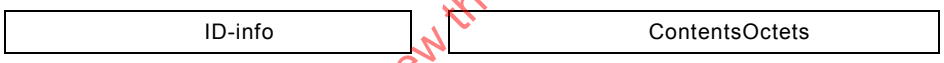


Figure 8 – Encoding of a context specific tagged value

5.3.2.2 Encoding of the identification information (ID-info)

The ID-info is composed of three fields: tag, structure flag and length field (see Figure 9).

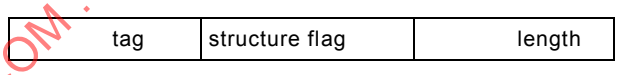


Figure 9 – Identification information fields

The length field contains either the number of the subsequent data octets or the number of the following values, as defined in this subclause.

If the structure flag is set to 1, it indicates that a structure follows. Within the structure encoding each value is identified by its own ID-info. In this case, the number of the values within the structure is encoded in the length field of the ID-info. In the length field of the last value of the structure, the length of the values encoding in octets is contained.

If the structure flag is 0, and the value is not part of a structure as described before, the following values are not tagged. In this case the types and lengths of the contained values are known implicitly; the number of the contained values(parameters) is encoded in the length field.

If the structure flag is 0, and the value is part of another structure, the length field shall contain:

the number of the following values, if this value is not the last value within the structure,

the number of the value encoding octets, if this value is the last value within the structure.

The tag field contains the tag value defined within the ASN.1 syntax definition. If the tag value is < 15 and the length value is < 7, the ID-info shall be encoded using 1 octet (see Figure 10).

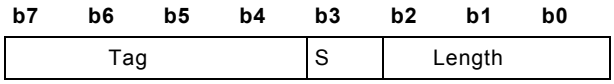


Figure 10 – ID-info for tag 0 .. 14 , length entry 0 .. 6

If the tag value is >= 15 and the length value is < 7, the ID-info shall be encoded using 2 octets (see Figure 11).

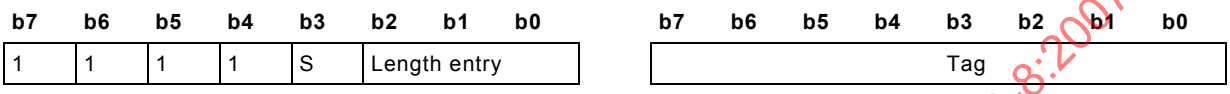


Figure 11 – ID-info for tag 15 .. 255 , length entry 0 .. 6

If the tag value is < 15 and the length value is >= 7, the ID-info shall be encoded using 2 octets (see Figure 12).

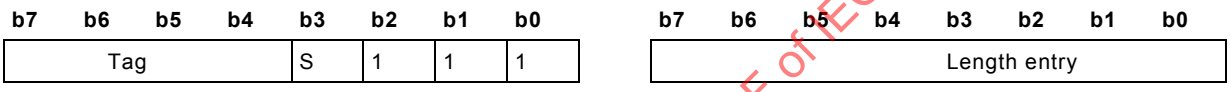


Figure 12 – ID-info for tag 0 .. 14 , length entry 7 .. 255

If the tag value is >= 15 and the length value is >= 7, the ID-info shall be encoded using 3 octets (see Figure 13).

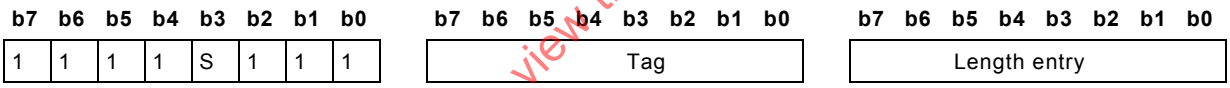


Figure 13 – ID-info for tag 15 .. 255 , length entry 7 .. 255

5.3.2.3 Example

Data of the data type Boolean, for example, is coded as shown in Figure 14 and Figure 15:

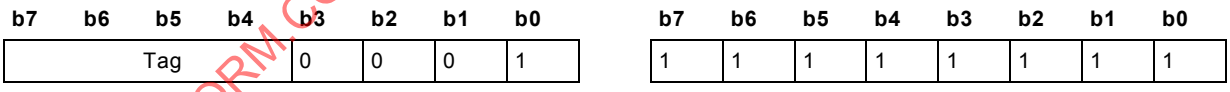


Figure 14 – Encoding of Boolean value TRUE

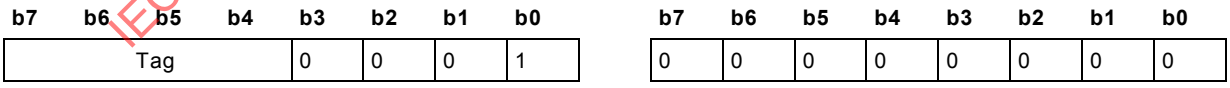


Figure 15 – Encoding of Boolean value FALSE

5.4 Encoding of simple values

5.4.1 Encoding of a Boolean value

1) The encoding of a Boolean value shall be primitive, and the ContentsOctets shall consist of exactly one octet.

2) The value TRUE is coded by setting all bits of the octet to 1, the value FALSE is coded by setting all bits of the octet to 0.

5.4.2 Encoding of String values (visible string, octet string, bit string)

In the encoding of strings, length entries, which give the number of following octets, are placed in the first octet of the individual string elements (see Figure 16). User data without a tag are provided with no ID-info. The data is identified by its position within the PDU and the length is implicitly known and fixed.

Octet 1	Octet 2	Octet 3		Octet n + 1
Length = n	Element 1	Element 2		Element n

Figure 16 – Encoding of Strings

5.4.3 Encoding of an Integer value

a) The encoding of a fixed-length Integer value of Integer8, Integer16, and Integer32 types shall be primitive, and the ContentsOctets shall consist of exactly one, two, or four octets, respectively.

b) The ContentsOctets shall be a two's complement binary number equal to the integer value, and consist of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each octet in turn up to and including the last octet of the ContentsOctets.

5.4.4 Encoding of an Unsigned value

a) The encoding of a fixed-length Unsigned value of Unsigned8, Unsigned16, and Unsigned32 types shall be primitive, and the ContentsOctets shall consist of exactly one, two, or four octets, respectively.

b) The ContentsOctets shall be a binary number equal to the Unsigned value, and consist of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each octet in turn up to and including the last octet of the ContentsOctets.

5.4.5 Encoding of a Floating-point value

a) The encoding of a Floating32 value shall be primitive, and the ContentsOctets shall consist of exactly four or eight octets, respectively.

b) The ContentsOctets shall contain floating-point values defined in conformance with ANSI/IEEE Standard 754. The sign is encoded in bit 8 of the first octet. It is followed by the exponent starting from bit 7 of the first octet, and then the mantissa starting from bit 7 of the second octet for Floating32.

5.4.6 Encoding of a BinaryDate value

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

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

c) The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 17.

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	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 min
4	SU	0	0	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…23 hours
	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	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…12 months
7	0	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0 … 99 years
	msb								

Figure 17 – Encoding of BinaryDate value

5.4.7 Encoding of a BinaryDate2000 value

a) The encoding of a BinaryDate2000 value shall be primitive.

b) The Length field shall indicate as a binary number the number of octets in the BinaryDate2000 value.

c) The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 18.

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	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 min
4	SU	0	0	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…23 hours
	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	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…12 months
7	0	0	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	0 … 9999 years
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	with century
	msb								

Figure 18 – Encoding of BinaryDate2000 value

5.4.8 Encoding of a Time-of-day value

a) The encoding of a Time Of Day value shall be primitive.

b) The Length field shall indicate as a binary number the number of octets in the Time Of Day value.

c) The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 19.

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 01.01.84
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	optional
	msb								

Figure 19 – Encoding of Time-of-day value

5.4.9 Encoding of a Time-difference value

a) The encoding of a Time Difference value shall be primitive.

b) The Length field shall indicate as a binary number the number of octets in the Time Difference value.

c) The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 20.

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 20 – Encoding of Time-difference value

5.4.10 Encoding of a Time value

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

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

c) The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 21.

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}	signed integer
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	of 8 octet length of 1/32ms unit
	MSB								

Figure 21 – Encoding of Time value

5.4.11 Encoding of a Null value

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

b) The ContentsOctet shall be omitted.

5.4.12 Encoding of an ANY value

The Data Type ANY contains one or more data elements of the Data Types which are chained together without any gap. The composition is known implicitly.

5.4.13 Encoding of structured types

When a structured type is encoded, the structure flag of the Identification Information shall be set to one. The Length field, whether or not it is extended, shall contain the number of components of the structured type being encoded.

5.4.14 Encoding of a SEQUENCE value

The SEQUENCE type is comparable to a record. It represents a collection of user data of the same or of different Data Types.

A SEQUENCE type value may contain a simple variable or a further structured variable as its components. If a SEQUENCE type contains another structured type value, it shall be counted as a single component even if it contains several components.

5.4.15 Encoding of a SEQUENCE OF value

The SEQUENCE OF type represents a succession of components. It is comparable to an array.

A SEQUENCE OF type value may contain one or more simple or constructed variables. If a SEQUENCE OF type contains another structured type value, it shall be counted as a single component even if it contains several components.

The encoding is as for the structure SEQUENCE. For the statement of the number of components the number of repetitions shall be taken into account.

5.4.16 Encoding of a CHOICE value

A CHOICE type represents a selection from a set of predefined values. 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 shall be encoded for a CHOICE.

5.4.17 Encoding of an ENUMERATED value

An ENUMERATED value shall be encoded as an Unsigned8 value.

5.4.18 Encoding of an Object-definition value

The encoding of the individual object definitions onto the parameters "List Of Objects and Attributes" (data type Gn_ObjectDefinition) of the GetAttributePDUs is shown in this subclause.

The objectClass attribute is the identifier of the object and indicates the class to which this object belongs. The other object attributes are object specific and shall be encoded as a string of octets. In addition to the object attributes, the object class shall be transmitted in the GetAttributes service, except List Header, whose numeric ID is zero. An example of the structure of an Object Definition is shown in Figure 22.

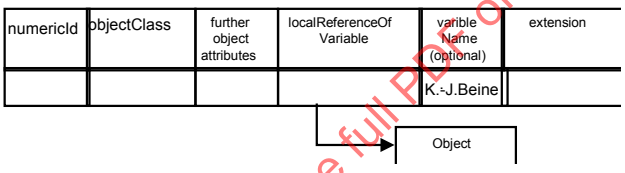


Figure 22 – Example for an Object definition

6 Protocol machine overview

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

NOTE The state machines specified in this subclause and ARPMs defined in the following clauses only define the valid events for each. It is a local matter to handle 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). Figure 23 shows the primitives exchanged between protocol machines.

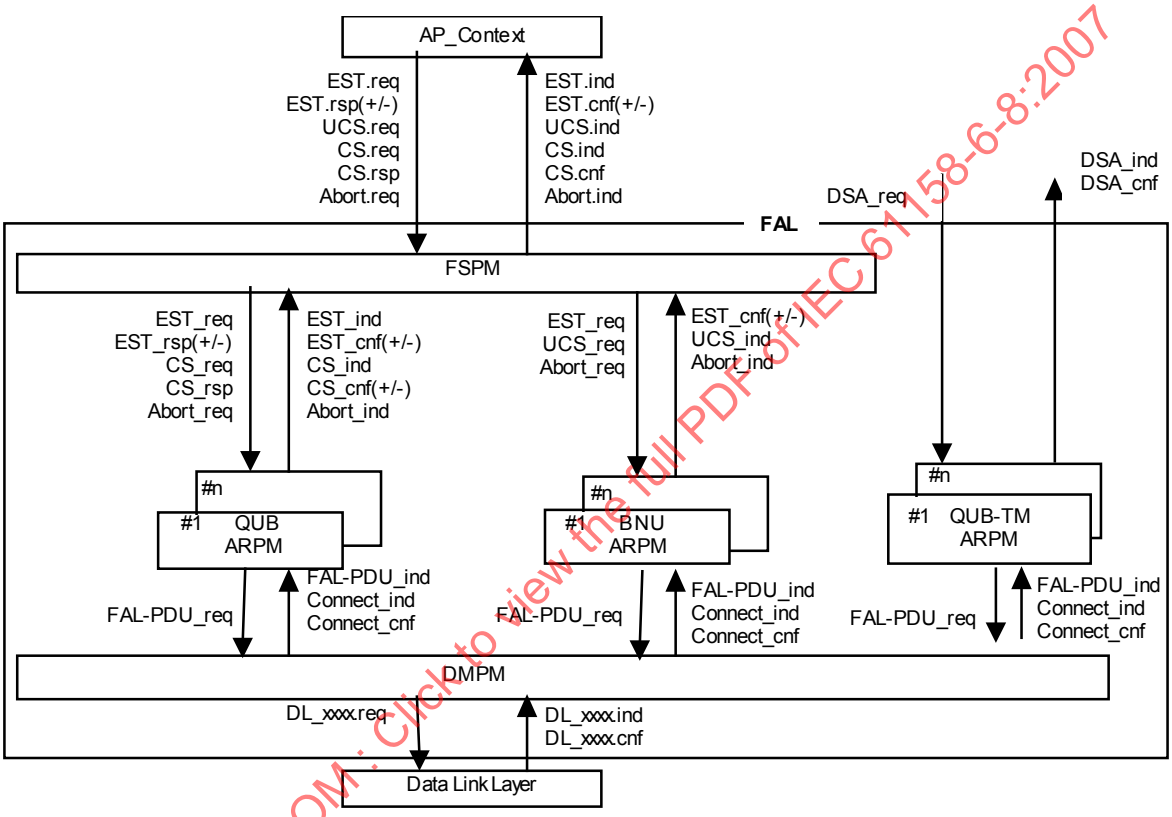


Figure 23 – Primitives exchanged between protocol machines

7 AP-context state machine

7.1 Primitive definitions

7.1.1 Primitives exchanged between FAL-user and AP-Context

Primitives exchanged between FAL-user and AP-Context are shown in Table 1 and Table 2.

Table 1 – Primitives issued by FAL-user to AP-context

Primitive name	Source	Associated parameters and functions
Terminate.req	FAL-User	Refer to FAL service definition (61158-508)
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 2 – Primitives issued by AP-context to FAL-user

Primitive name	Source	Associated parameters and functions
Terminate.ind	AP-Context	Refer to FAL service definition (61158-508)
Initiate.ind	AP-Context	
Initiate.cnf(+)	AP-Context	
Initiate.cnf(-)	AP-Context	
ConfirmedService.ind	AP-Context	
ConfirmedService.cnf	AP-Context	
UnconfirmedService.ind	AP-Context	

7.2 State machine description

The attributes used in this state machine are defined as elements of the AP attribute *List Of In-Use AR Endpoint Info*. See Figure 24 for the state machine.

CLOSED

The AP-Context for an AR is not open. Only the service primitives Initiate.req, Establish.ind, Terminate.req and Abort.ind are allowed. All other service primitives shall be rejected with the Terminate service.

OPENING-REQUESTING (REQ)

The local FAL user wishes to open the AP-Context for an AR. Only the service primitives Establish.cnf(+), Establish.cnf(-), Terminate.req and Abort.ind are allowed. All other services shall be rejected with the Terminate service.

OPENING-RESPONDING (RSP)

The remote AP-Context wishes to open the AP-Context for an AR. Only the service primitives Initiate.rsp(+), Initiate.rsp(-), Terminate.req and Abort.ind are allowed. All other service primitives shall be rejected with the Terminate service.

OPEN

The AP-Context for an AR is open. The service primitives Initiate.req, Initiate.rsp(+), Initiate.rsp(-) are not allowed and shall be rejected with the Terminate service. The following actions shall be taken to reset the AP-Context for an AR.

Set the *Outstanding Services Requesting Counter* and *Outstanding Services Responding Counter* to 0.

Set the *Context State* to "CLOSED"

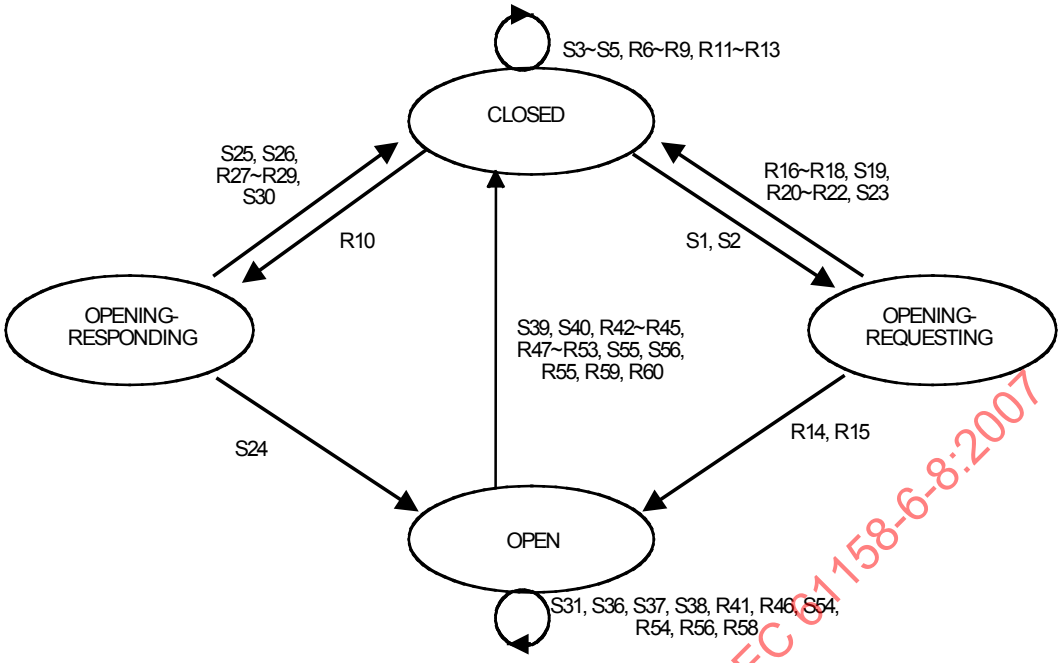


Figure 24 – AP to AP-context initiation state machine

7.3 AP to AP-context initiation state transitions

The AP to AP-Context initiation state transitions are defined in Table 3 and Table 4. In the tables of this subclause "Data" parameter for FAL Service Protocol Machine is expressed by "FalApduBody" to emphasize its semantics.

Table 3 – AP-context state machine sender transactions

#	Current state	Event or condition => action	Next State

S1	CLOSED	Initiate.req
		&& ApContextTest = "True"
		&& ApExplicitConnection = "True"
		=>
		EST.req {
		FalApduBody := "Establish-CommandPDU"
		}
		or
		EST.req {
		FalApduBody := "Establish-RequestPDU"
		}
		or
		EST.req {
		FalApduBody := "Establish-Request2PDU"
		}
			REQ

S2	CLOSED	Initiate.req
		&& ApContextTest = "True"
		&& ApExplicitConnection = "False"
		=>
		EST.req {
		FalApduBody := "NULL"
		}
			REQ

S3	CLOSED	Initiate.req
		&& ApContextTest = ""False""
		=>
		Terminate.ind {
		LocallyGenerated := "True",
		TerminateIdentifier := "FAL",
		ReasonCode := "Context error"
		}
			CLOSED

#	Current state	Event or condition => action	Next State
S4	CLOSED	Terminate.req => (no actions taken)	CLOSED

S5	CLOSED	FAL service.primitive <> “Initiate”
		&& FAL service.primitive <> “Terminate”
		=>
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “User error”
		}
			CLOSED

S19	REQ	Terminate.req => Abort.req { Originator:= “TerminateIdentifier of Terminate.req”, ReasonCode := “ReasonCode of Terminate.req” }, ResetArep	CLOSED

S23	REQ	FAL service.primitive <> “Initiate.rsp””
		&& FAL service.primitive <> “Terminate.req”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “User error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “User error”
		},
		
		ResetArep
			CLOSED

S24	RSP	Initiate.rsp(+)
		=>
		EST.rsp(+) {
		FalApduBody := “Establish-AffirmativePDU”
		}
		or
		EST.rsp(+) {
		FalApduBody = “Establsih-ResponsePDU”
		}
			OPEN

S25	RSP	Initiate.rsp(-)
		=>
		EST.rsp(-){
		FalApduBody := “Establish-NegativePDU”
		},
		or
		EST.rsp(-) {
		FalApduBody = “Establsih-ErrorPDU”
		},
		
		ResetArep
			CLOSED

S26	RSP	Terminate.req => Abort.req { Originator := “TerminateIdentifier of Terminate.req”, ReasonCode := “ReasonCode of Terminate.req” }, ResetArep	CLOSED

#	Current state	Event or condition => action	Next State

S30	RSP	FAL service.primitive <> “Initiate.rsp”
		&& FAL service.primitive <> “Terminate.req”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “User error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “User error”
		},
		
		ResetArep
			CLOSED

S31	OPEN	ConfirmedService.req && ConfirmedServiceCheck = “True” && OutstandingServicesRequestingCounter < NegotiatedMaxOutstanding-ServicesRequesting && InvokeIdExistent = “False” && PDU length ≤ NegotiatedMaxPduSizeSending && RequestedServiceSupportedTest = “True” => CS.req { FalApduBody := “ConfirmedSend-CommandPDU” }, or CS.req { FalApduBody := “ConfirmedSend-RequestPDU” }, OutstandingServicesRequestingCounter := OutstandingServicesRequestingCounter+1	OPEN

S36	OPEN	UnconfirmedService.req
		&& UnconfirmedServiceCheck = “True”
		&& PDU length ≤ NegotiatedMaxPduSizeSending
		&& RequestedServiceSupportedTest = “True”
		&& ImmediateAcknowledge = “False”
		=>
		UCS.req {
		FalApduBody := “UnconfirmedSend-CommandPDU”
		}
		or
		UCS.req {
		FalApduBody := “UnconfirmedSend-PDU”
		}
			OPEN

S37	OPEN	UnconfirmedService.req
		&& UnconfirmedServiceCheck = “True”
		&& PDU length ≤ NegotiatedMaxPduSizeSending
		&& RequestedServiceSupportedTest = “True”
		&& ImmediateAcknowledge = “True”
		=>
		UCA.req {
		FalApduBody := “UnconfirmedAcknowledgedSend-CommandPDU”
		}
		or
		UCA.req {
		FalApduBody := “UnconfirmedAcknowledgedSend-RequestPDU”
		}
			OPEN

S38	OPEN	AR_ASE service request && ArServiceCheck = “True” && RequestedServiceSupportedTest = “True” => ArFspmService NOTE This state is provided to access the FSPM direct from the FAL User. The function ArFspmService generates an FSPM primitive as defined later in this subclause.	OPEN

#	Current state	Event or condition => action	Next State
S39	OPEN	Terminate.req => Abort.req { Originator := “TerminateIdentifier of Terminate.req”, ReasonCode := “ReasonCode of Terminate.req” }, ResetArep	CLOSED

S40	OPEN	Faulty, unknown or not-allowed FAL service.primitive
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “User error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “User error”
		},
		
		ResetArep
			CLOSED

S54	OPEN	ConfirmedService.rsp && ConfirmedServiceCheck = “True” && InvokeIdExistent = “True” && SameService = “True” && PDU length ≤ NegotiatedMaxPduSizeSending => CS.rsp { FalApduBody := “ConfirmedSend-AffirmativePDU” }, or CS.rsp { FalApduBody := “ConfirmedSend-NegativePDU” }, or CS.rsp { FalApduBody := “ConfirmedSend-ResponsePDU” }. OutstandingServicesRespondingCounter := OutstandingServicesRespondingCounter-1 NOTE An extra protocol means shall be provided to detemine whether ConfirmedSend-AffirmativePDU or ConfirmedSend-NegativePDU is used.	OPEN

S55	OPEN	ConfirmedService.rsp
		&& ConfirmedServiceCheck = “True”
		&& InvokeIdExistent = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “InvokeID-error-response”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-response”
		},
		
		ResetArep
			CLOSED

#	Current state	Event or condition => action	Next State

S56	OPEN	ConfirmedService.rsp
		&& ConfirmedServiceCheck = “True”
		&& InvokeIdExistent = “True”
		&& SameService = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Service-error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Service-error”
		},
		
		ResetArep
			CLOSED

Table 4 – AP-context state machine receiver transactions

#	Current state	Event or condition => action	Next state
R6	CLOSED	Abort.ind => (no actions taken)	CLOSED

R7	CLOSED	Faulty or unknown AR_FSPM service primitive
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR_ASE error”
		}
			CLOSED

R8	CLOSED	AR_FSPM service primitive <> “EST.ind”
		&& AR_FSPM service.primitive <> “Abort.ind”
		=>
		Abort.req {
		Originator:= “FAL”,
		ReasonCode := “Connection State Conflict”
		}
			CLOSED

R9	CLOSED	EST.ind
		&& FalApduBody = not allowed, unknown or faulty FAL PDU
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “FAL PDU error”
		}
			CLOSED

R10	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “True”
		&& MaxFalPduLengthTest = "True"
		&& ServicesSupportedTest = “True”
		=>
		NegotiateOutstandingServices,
		
		Initiate.ind {
		}
			RSP

R11	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR error”
		}
			CLOSED

#	Current state	Event or condition => action	Next state

R12	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “True”
		&& MaxFalPduLengthTest = “False”
		=>
		EST.rsp(-) {
		FalApduBody := “Establish-NegativePDU”,
		ErrorCode := “Max-Fal-Fdu-Size-Insufficient”
		}
		or
		EST.rsp(-) {
		FalApduBody := “Establish-ErrorPDU”,
		ErrorCode := “Max-Fal-Fdu-Size-Insufficient”
		}
			CLOSED

R13	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “True”
		&& MaxFalPduLengthTest = “True”
		&& ServicesSupportedTest = “False”
		=>
		EST.rsp(-) {
		FalApduBody := “Establish-NegativePDU”,
		ErrorCode := “Service-Not-Supported”
		}
		or
		EST.rsp(-) {
		FalApduBody := “Establish-ErrorPDU”,
		ErrorCode := “Service-Not-Supported”
		}
			CLOSED

R14	REQ	EST.cnf(+)
		&& (FalApduBody = “Establish-AffirmativePDU”
		|| FalApduBody = “Establish-ResponsePDU”)
		&& ApExplicitConnection = “True”
		=>
		Initiate.cnf(+) {
		}
			OPEN

R15	REQ	EST.cnf(+)
		&& FalApduBody = “NULL”
		&& ApExplicitConnection = “False”
		=>
		Initiate.cnf(+) {
		}
			OPEN

R16	REQ	EST.cnf(-) && (FalApduBody = “Establish-NegativePDU” || FalApduBody = “Establish-ErrorPDU”) && ApExplicitConnection = “True” => “Initiate.cnf(-) { ErrorCode := “Errorcode of EST.cnf(-)” }, ResetArep	CLOSED
R17	REQ	EST.cnf(-) && FalApduBody = “NULL” && ApExplicitConnection = “False” => Initiate.cnf(-) { ErrorCode := “Errorcode in EST.cnf” }, ResetArep	CLOSED

#	Current state	Event or condition => action	Next state
R18	REQ	Abort.ind => Terminate.ind { LocallyGenerated := “LocallyGenerated of Abort.ind”, TerminateIdentifier := “Originator of Abort.ind”, ReasonCode := “ReasonCode of Abort.ind” }, ResetArep	CLOSED

R20	REQ	Faulty or unknown AR_FSPM service primitive
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		=>
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		ResetArep
			CLOSED

R21	REQ	AR_FSPM service.primitive <> “EST.cnf”
		&& AR_FSPM service.primitive <> “Abort.ind”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Connection State Conflict”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Connection State Conflict”
		},
		
		ResetArep
			CLOSED

R22	REQ	EST.cnf
		&& FalApduBody =“ not allowed, unknown or faulty FAL PDU”
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “FAL PDU error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “FAL PDU error”
		},
		
		ResetArep
			CLOSED

R27	RSP	Abort.ind => Terminate.ind { LocallyGenerated := “LocallyGenerated of Abort.ind”, TerminateIdentifier := “Originator of Abort.ind”, ReasonCode := “ReasonCode of Abort.ind” }, ResetArep	CLOSED

#	Current state	Event or condition => action	Next state

R28	RSP	Faulty or unknown AR_FSPM service primitive
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		ResetArep
			CLOSED

R29	RSP	AR_FSPM service primitive <> “ Abort.ind”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “State Conflict with AR_ASE”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “AP_ASE”,
		ReasonCode := “Connection State Conflict with AR_ASE”
		},
		
		ResetArep
			CLOSED

R41	OPEN	CS.ind && (FalApduBody = “ConfirmedSend-CommandPDU” || FalApduBody = “ConfimredSend-RequestPDU”) && PDU length ≤ NegotiatedMaxPduSizeReceiving && OutstandingServicesRespondingCounter < NegotiatedMaxOutstanding-ServicesResponding && InvokeIdExistent = “False” && IndicatedServiceSupportedTest = “True” => ConfirmedService.ind { }, OutstandingServicesRespondingCounter := OutstandingServicesRespondingCounter+1	OPEN

R42	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

#	Current state	Event or condition => action	Next state

R43	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& OutstandingServicesResponding ≥ NegotiatedMaxOutstanding-ServicesResponding
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Max-services-overflow”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Max-Services-Overflow”
		},
		
		ResetArep
			CLOSED

R44	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& OutstandingServicesRespondingCounter < NegotiatedMaxOutstanding-ServicesResponding
		&& InvokeIdExistent = “True”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “InvokeID-error-request”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-request”
		},
		
		ResetArep
			CLOSED

R45	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& OutstandingServicesRespondingCounter < NegotiatedMaxOutstandingServicesResponding
		&& InvokeIdExistent = “False”
		&& IndicatedServiceSupportedTest = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		ResetArep
			CLOSED

R46	OPEN	UCS.ind
		&& (FalApduBody = “UnconfirmedSend-CommandPDU”
		|| FalApduBody = “UnconfimredSend-PDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& IndicatedServiceSupportedTest = “True”
		=>
		UnconfirmedService.ind {
		}
			OPEN

#	Current state	Event or condition => action	Next state

R47	OPEN	UCS.ind
		&& (FalApduBody = “UnconfirmedSend-CommandPDU”
		|| FalApduBody = “UnconfimredSend-PDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

R48	OPEN	UCS.ind
		&& (FalApduBody = “UnconfirmedSend-CommandPDU”
		|| FalApduBody = “UnconfimredSend-PDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& IndicatedServiceSupportedTest = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		ResetArep
			CLOSED

R49	OPEN	Abort.ind => Terminate.ind { LocallyGenerated := “LocallyGenerated of Abort.ind”, TerminateIdentifier := “Originator of Abort.ind”, ReasonCode := “ReasonCode of Abort.ind” }, ResetArep	CLOSED

R50	OPEN	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		ResetArep
			CLOSED

#	Current state	Event or condition => action	Next state

R51	OPEN	Faulty or unknown AR_FSPM service primitive
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		ResetArep
			CLOSED

R52	OPEN	Not-allowed AR_FSPM service primitive
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		ResetArep
			CLOSED

R53	OPEN	valid AR_FSPM Send Service primitive (one of CS, US, UCA)
		&& FalApduBody = not-allowed, unknown or faulty FAL PDU
		&& ArAccessSupported = “False”
		=>
		Abort.req {
		Originator := “AP_ASE”,
		ReasonCode := “FAL-PDU-error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “FAL –PDU-error”
		},
		
		ResetArep
			CLOSED

R54	OPEN	UCA.ind
		&& ArAccessSupported = “True”
		&& (FalApduBody = “UnconfirmedAcknowledgedSend-CommandPDU”
		|| FalApduBody = “UnconfirmedAcknowledgedSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		=>
		UnconfirmedService.ind {
		}
			OPEN

R55	OPEN	UCA.ind
		&& ArAccessSupported = “True”
		&& (FalApduBody = “UnconfirmedAcknowledgedSend-CommandPDU”
		|| FalApduBody = “UnconfirmedAcknowledgedSend-RequestPDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

#	Current state	Event or condition => action	Next state
R56	OPEN	AR_ASE service indication => ArFspmService NOTE This state is provided to access the FSPM direct from the FAL User. The function ArFspmService generates an FSPM primitive as defined later in this subclause.	OPEN
R58	OPEN	CS.cnf && (FalApduBody = “ConfirmedSend-AffirmativePDU” || FalApduBody = “ConfirmedSend-NegativePDU” || FalApduBody = “ConfirmedSend-ResponsePDU”) && PDU length ≤ NegotiatedMaxPduSizeReceiving && InvokeIdExistent = “True” && SameService = “True” => ConfirmedService.cnf { }, OutstandingServicesRequestingCounter := OutstandingServicesRequestingCounter-1	OPEN

R59	OPEN	CS.cnf
		&& (FalApduBody = “ConfirmedSend-AffirmativePDU”
		|| FalApduBody = “ConfirmedSend-NegativePDU”
		|| FalApduBody = “ConfirmedSend-ResponsePDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

R60	OPEN	CS.cnf
		&& (FalApduBody = “ConfirmedSend-AffirmativePDU”
		|| FalApduBody = “ConfirmedSend-NegativePDU”
		|| FalApduBody = “ConfirmedSend-ResponsePDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& InvokeIdExistent = “False”
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-responding”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-responding”
		},
		
		ResetArep
			CLOSED

7.4 Functions

Table 5 through Table 20 define internal functions that are used by the AP-Context state machine.

Table 5 – Function ResetArep

Name	ResetArep		
Input	**Arep_Id**	**Output**	**True or False**
Function	Closes the AP AR Context and initializes all elements of the AP attribute *List Of In-Use AR Endpoint Info* to zero (0).		

Table 6 – Function ApContextTest

Name	ApContextTest		
Input	**Arep_Id**	**Output**	**True or False**
Function	Locates the entry for the Selected AREP in the AP attribute *List Of In-Use AR Endpoint Info*, verifies its contents, and ensures that there is a matching AREP defined for the AR_ASE. The way in which the verification is carried out is dependent on implementation.		

Table 7 – Function ServicesSupportedTest

Name	ServicesSupportedTest		
Input	**Arep_Id**	**Output**	**True or False**
Function	Upon receipt of an Initiate-RequestPDU the called AP_ASE shall compare the Local List of Supported Services against those contained in the Initiate PDU. The ServicesSupportedTest fails if either the Local FAL does not support all the services as a responder that the remote FAL supports as a requestor, or the remote FAL does not support all the services as a responder that the local FAL supports as a requestor.		

Table 8 – Function ApExplicitConnection

Name	ApExplicitConnection		
Input	**Arep_Id**	**Output**	**True or False**
Function	Refer to AREP to determine if this AP_ASE supports explicit connection between APs.		

Table 9 – Function ImmediateAcknowledge

Name	ImmediateAcknowledge		
Input	**Arep_Id**	**Output**	**True or False**
Function	Refer to AREP to determine if this AP_ASE requires immediate acknowledge in the underlying layer.		

Table 10 – Function ConfirmedServiceCheck

Name	ConfirmedServiceCheck		
Input	**Arep_Id**	**Output**	**True or False**
Function	Determines if the called FAL service is a confirmed service except for AR ASE services.		

Table 11 – Function UnconfirmedServiceCheck

Name	UnconfirmedServiceCheck		
Input	**Arep_Id**	**Output**	**True or False**
Function	Determines if the called FAL service is an unconfirmed service except AR ASE services.		

Table 12 – Function ArServiceCheck

Name	ArServiceCheck		
Input	**Arep_Id**	**Output**	**True or False**
Function	Determines if the called service is an AR ASE service (FAL service or AR FSPM service).		

Table 13 – Function ArFspmService

Name	ArFspmService		
Input	**Arep_Id**	**Output**	**FAL Service or AR FSPM Service**
Function	Generate an AR ASE service primitive. Relationship between FAL Services and AR FSPM Services shall be as follows: AR-Unconfirmed Send UCS AR-Confirmed Send CS AR-Establish EST AR-Abort Abort		

Table 14 – Function ArAcceeSupported

Name	ArAcceeSupported		
Input	**Arep_Id**	**Output**	**True or False**
Function	Determines if the AP accepts AR ASE Send (ConfirmedSend or UnconfirmedSend) services.		

Table 15 – Function MaxFalPduLengthTest

Name	MaxFalPduLengthTest		
Input	**Arep_Id**	**Output**	**True or False**
Function	Upon receipt of an Initiate-RequestPDU, the called AP_ASE shall check the requested maximum PDU length against its defined context. The following tests are performed: 1. If (ConfiguredMaxPDUSizeSending > MaxPDUSizeReceiving of the Initiate PDU), this test fails; 2. If (ConfiguredMaxPDUSizeReceiving < MaxPDUSizeSending of the Initiate PDU), this test fails.		

Table 16 – Function NegotiateOutstandingServices

Name	NegotiateOutstandingServices		
Input	**Arep_Id**	**Output**	**True or False**
Function	Upon receipt of an Initiate-RequestPDU, the called AP_ASE shall perform the following negotiation: 1. If ConfiguredMaxOustandingServicesRequesting > MaxOutstandingServicesResponding of the Initiate-RequestPDU then NegotiatedMaxOutstandingServicesRequesting = MaxOutstandingServicesResponding of the Initiate-RequestPDU else NegotiatedMaxOutstandingServicesRequesting = ConfiguredMaxOutstandingServicesRequesting; 2. If ConfiguredMaxOutstandingServicesResponding < MaxOutstandingServicesRequesting of the Initiate PDU then NegotiatedMaxOutstandingServicesResponding = ConfiguredMaxOutstandingServicesResponding else NegotiatedMaxOutstandingServicesResponding = MaxOutstandingServicesRequesting of the Initiate PDU;		

Table 17 – Function RequestedServicesSupportedTest

Name	RequestedServicesSupportedTest		
Input	**Arep_Id, Service.primitive**	**Output**	**True or False**
Function	Upon receipt of a service request from FAL user, the called AP_ASE shall compare the Local List of Supported Services against the requested service primitive. This test fails if the service primitive is not contained in the Local List of Supported Services.		

Table 18 – Function IndicatedServicesSupportedTest

Name	IndicatedServicesSupportedTest		
Input	**Arep_Id, service.primitive**	**Output**	**True or False**
Function	Upon receipt of a service indication from AR_ASE, the called AP_ASE shall compare the Local List of Supported Services against the indicated service primitive. This test fails if the service primitive is not contained in the Local List of Supported Services.		

Table 19 – Function InvokeIdExistent

Name	InvokeIdExistent		
Input	**Arep_Id, Invoke_Id**	**Output**	**True or False**
Function	Upon receipt of a service primitive, the called AP_ASE shall 1. compare the local list of invoke IDs in use against the requested Invoke ID if the service primitive is request; 2. compare the local list of invoke IDs in use against the responded invoke ID if the service primitive is response. This test fails if the invoke ID does not exist in the local list of Invoke IDs in use.		

Table 20 – Function SameService

Name	SameService		
Input	**Arep_Id, Invoke_Id**	**Output**	**True or False**
Function	Upon receipt of a service primitive the called AP_ASE shall 1. compare the local list of outstanding services (responding) against the service if the service primitive is response; 2. compare the local list of outstanding services (requesting) against the service if the service primitive is confirmation. This test fails if the service is not the same as that in the local list of outstanding services.		

8 FAL service protocol machine (FSPM)

8.1 Summary

The FAL 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."

8.2 Primitive definitions

8.2.1 Primitives exchanged between AP-context and FSPM

The primitives exchanged between the AP-Context and the FSPM are described in Table 21 and Table 22.

Table 21 – Primitives issued by AP-context to FSPM

Primitive name	Source	Associated parameters	Functions
EST.req	AP-Context	Arep_Id, Data, Remote_DLCEP_Address	This primitive is used to convey an Establish request primitive from the AP-Context to the FSPM.
EST.rsp(+)	AP-Context	Arep_Id, Data	This primitive is used to convey an Establish response(+) primitive from the AP-Context to the FSPM.
EST.rsp(-)	AP-Context	Arep_Id, Data	This primitive is used to convey an Establish response(-) primitive from the AP-Context to the FSPM.
Abort.req	AP-Context	Arep_Id, Identifier, Reason_Code, Additional_Detail	This primitive is used to convey an Abort request primitive from the AP-Context to the FSPM.
CS.req	AP-Context	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) request primitive from the AP-Context to the FSPM.
CS.rsp	AP-Context	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) response primitive from the AP-Context to the FSPM.
UCS.req	AP-Context	Arep_Id, Remote_DLSAP_Address, Data	This primitive is used to convey an Unconfirmed Send (UCS) request primitive from the AP-Context to the FSPM.
UCA.req	AP-Context	Arep_Id, Remote_Dlsap_Address, Data	This primitive is used to send an unconfirmed service request.

Table 22 – Primitives issued by FSPM to AP-context

Primitive name	Source	Associated parameters	Functions
EST.ind	FSPM	Arep_Id, Data	This primitive is used to convey an Establish indication primitive from the FSPM to the AP-Context.
EST.cnf(+)	FSPM	Arep_Id, Data	This primitive is used to convey an Establish result(+) primitive from the FSPM to the AP-Context.
EST.cnf(-)	FSPM	Arep_Id, Data	This primitive is used to convey an Establish result(-) primitive from the FSPM to the AP-Context.
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 AP-Context.
CS.ind	FSPM	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) indication primitive from the FSPM to the AP-Context.
CS.cnf	FSPM	Arep_Id, Data	This primitive is used to convey a Confirmed Send (CS) confirmation primitive from the FSPM to the AP-Context.
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 AP-Context.
UCA.ind	FSPM	Arep_Id, Remote_DLSAP_Address, Duplicate_FAL-SDU, Data	This primitive is used to convey an unconfirmed service indication.

8.2.2 Parameters of AP-context /FSPM primitives

All the parameters used in the primitives exchanged between the AP-Context and the FSPM are identical to those defined in the “Operational Services” subclause.

8.3 FSPM state tables

8.3.1 General

The FSPM state machines are described in Figure 25, Table 23 and Table 24.

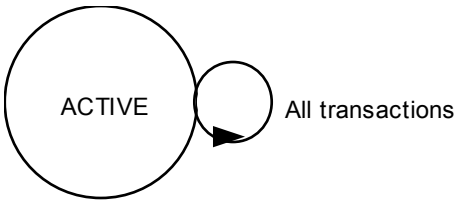


Figure 25 – State transition diagram of FSPM

Table 23 – FSPM state table – sender transactions

#	Current state	Event or condition => action	Next state

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

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

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

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

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

S6	ACTIVE	CS.rsp
		&& SelectArep (Arep_Id) = "True"
		=>
		CS_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

#	Current state	Event or condition => action	Next state

S12	ACTIVE	UCA.req
		&& SelectArep (Arep_Id) = "True"
		=>
		UCA_req {
		remote_dlsap_address := Remote_DLSAP_Address,
		user_data := Data
		}
			ACTIVE

S14	ACTIVE	Any FSPM.req && SelectArep (Arep_Id) = “False” => (no actions defined by the protocol, see notes 1 and 2.)	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 AP-Context 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 for the ARPM.

Table 24 – FSPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	ACTIVE	EST_ind
		=>
		EST.ind {
		Arep_Id := arep_id,
		Data := user_data
		}
			ACTIVE

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

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

R4	ACTIVE	UCS_ind
		=>
		UCS.ind {
		Arep_Id := arep_id,
		Remote_DLSAP_Address := remote_dlsap_address,
		Data := user_data,
		}
			ACTIVE

R5	ACTIVE	CS_ind
		=>
		CS.ind {
		Arep_Id := arep_id,
		Data := user_data
		}
			ACTIVE

R6	ACTIVE	CS_cnf
		=>
		CS.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

R16	ACTIVE	UCA_ind
		=>
		UCA.ind {
		Arep_Id := arep_id,
		Remote_DLSAP_Address := remote_dlsap_address,
		Data := user_data,
		}
			ACTIVE

8.3.2 Functions

The function used in this state machine is as shown in Table 25.

Table 25 – 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 AP-Context service primitives.			

9 Application relationship protocol machines (ARPMs)

9.1 Queued user-triggered bidirectional-flow control (QUB-FC) ARPM

9.1.1 QUB-FCPrimitive definitions

9.1.1.1 Primitives exchanged between ARPM and FSPM

Table 26 and Table 27 list the primitives exchanged between the ARPM and the FSPM.

Table 26 – Primitives issued by FSPM to ARPM

Primitive name	Source	Associated parameters	Functions
EST_req	FSPM	user_data	This is an FAL internal primitive used to convey an Establish request primitive from the FSPM to the ARPM.
EST_rsp(+)	FSPM	user_data	This is an FAL internal primitive used to convey an Establish response(+) primitive from the FSPM to the ARPM.
EST_rsp(-)	FSPM	user_data	This is an FAL internal primitive used to convey an Establish response(-) primitive from the FSPM to the ARPM.
Abort_req	FSPM	identifier, reason_code, additional_detail	This an FAL internal primitive used to convey an Abort request primitive from the FSPM to the ARPM.
CS_req	FSPM	user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) request primitive from the FSPM to the ARPM.
CS_rsp	FSPM	user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) response primitive from the FSPM to the ARPM.
UCA_req	FSPM	remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Acknowledged Send (UCA) request primitive from the FSPM to the ARPM.

Table 27 – Primitives issued by ARPM to FSPM

Primitive name	Source	Associated parameters	Functions
EST_ind	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish indication primitive from the ARPM to the FSPM.
EST_cnf(+)	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish response(+) primitive from the ARPM to the FSPM.
EST_cnf(-)	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish response(-) primitive from the ARPM to the FSPM.
Abort_ind	ARPM	arep_id, locally_generated, identifier, reason_code, additional_detail	This is an FAL internal primitive used to convey an Abort primitive from the ARPM to the FSPM.
CS_ind	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) indication primitive from the ARPM to the FSPM.
CS_cnf	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey a Confirmed Send (CS) confirmation primitive from the ARPM to the FSPM.
UCA_ind	ARPM	arep_id, remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Acknowledged Send (UCA) indication primitive from the ARPM to the FSPM.

9.1.2 Parameters of FSPM/ARPM primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in Table 28.

Table 28 – Parameters used with primitives exchanged between FSPM and ARPM

Parameter name	Description
arep_id	This parameter is used to unambiguously identify an instance of the AREP that has issued a primitive. A means for such identification is not specified by this standard.
User_data	This parameter conveys FAL-User data.
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.

9.1.3 DLL mapping of QUB-FC AREP Class

This subclause describes the mapping of the QUB-FC AREP class to the Type 8 Data Link Layer defined in IEC 61158-3-8 and IEC 61158-4-8. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the Data Link Layer standard; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes are not in the scope of this International Standard.

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

CLASS: **Type 8 QubFC**
PARENT CLASS: **QueuedUser-TriggeredBidirectional-FlowControlAREP**
ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	QosParameterSet
3.1	(m)	Atribute:	DllPriorityNegotiated
3.2	(m)	Attribute:	MaxConfirmDelay
3.3	(m)	Attribute:	MaxDlsduSizes
3.3.1	(m)	Attribute:	MaxDlsduSizeFromRequestor
3.3.2	(m)	Attribute:	MaxDlsduSizeFromResponder
3.3.3	(m)	Attribute:	MaxDlsduSizeFromRequestorNegotiated
3.3.4	(m)	Atribute:	MaxDlsduSizeFromResponderNegotiated
3.4	(m)	Attribute:	MaxQueueDepth
3.4.1	(m)	Attribute:	MaxSendingQueueDepth
3.4.2	(m)	Attribute:	MaximimReceivingQueueDepth

DLL SERVICES:

1	(m)	OpsService:	DL-Data

9.1.4 Attributes

9.1.4.1 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.

9.1.4.2 RemoteDlcepAddress

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

9.1.4.3 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.

This attribute specifies the negotiated value of the DLL priority.

9.1.4.4 MaxConfirmDelay

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

9.1.4.5 MaxDlsduSize

MaxDlsduSizeFromRequestor

This attribute specifies the configured value of the maximum length of an FAL-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 responder” parameter of the DLL.

MaxDlsduSizeFromResponder

This attribute specifies the configured value of the maximum length of an FAL-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.

MaxDlsduSizeFromRequestorNegotiated

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

MaxDlsduSizeFromResponderNegotiated

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

9.1.4.6 MaxQueueDepth

MaxSendingQueueDepth

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

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

MaxReceivingQueueDepth

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

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

9.1.5 DLL services

Refer to IEC 61158-3-8 for DLL service descriptions.

9.1.6 QUB-FC ARPM state machine

9.1.6.1 QUB-FC ARPM states

The defined states and their descriptions of the QUB-FC ARPM are shown in Table 29 and Figure 26.

Table 29 – QUB-FC ARPM states

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

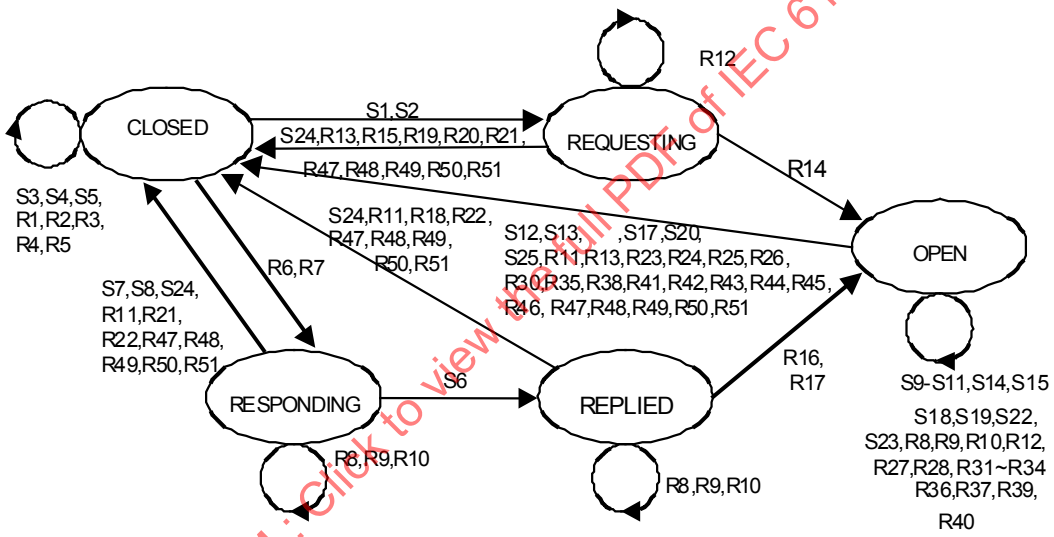


Figure 26 – State transition diagram of QUB-FC ARPM

9.1.6.2 QUB-FC ARPM state table

Table 30 and Table 31 define the state machine of the QUB-FC ARPM.

Table 30 – QUB-FC ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1	CLOSED	EST_req
		&& Initiator = “True”
		&& RemoteAddressConfigurationType := “Free”
		=>
		RemoteDlcepAddress := remote_dlcep_address,
		
		FAL-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 := BuildFAL-PDU (
		fal_pdu_name := “EST_ReqPDU”,
		calling_dlcep_address := LocalDlcepAddress,
		called_dlcep_address := RemoteDlcepAddress,
		fal_data := user_data)
		}
			REQ

S2	CLOSED	EST_req
		&& Initiator = “True”
		&& RemoteAddressConfigurationType := “Linked”
		=>
		FAL-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 := BuildFAL-PDU (
		fal_pdu_name := “EST_ReqPDU”,
		calling_dlcep_address := LocalDlcepAddress,
		called_dlcep_address := RemoteDlcepAddress,
		fal_data := user_data)
		}
			REQ

S3	CLOSED	EST_req
		&& Initiator = “False”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := “True”,
		identifier := “FAL”,
		reason_code := “Invalid Event for Role”
		}
			CLOSED

S4	CLOSED	unallowed primitive
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := “True”,
		identifier := “FAL”,
		reason_code := “Unallowed AREP primitive in connection establishment “
		}
			CLOSED

S5	CLOSED	Abort_req => ignore	CLOSED

S6	RSP	EST_rsp(+)
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Connect_rsp”,
		arep_id := GetArepId (),
		responding_address := “default dlsap address”,
		local_dlcep_address := LocalDlcep,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := “EST_RspPDU”,
		fal_data := user_data)
		}
			REPL

#	Current state	Event or condition => action	Next state

S7	RSP	EST_rsp(-)
		=>
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := "connection rejection-transient condition",
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "EST_ErrPDU",
		fal_data := user_data)
		}
			CLOSED

S8	RSP	unallowed primitive
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Unallowed AREP primitive in connection establishment "
		},
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := " Unallowed AREP primitive in connection establishment ",
		dlsdu := "null"
		}
			CLOSED

S9	OPEN	CS_req && Role = “Client” || “Peer” && GetCounterValue(OSCC) < maxOSCC && CIU = 0 => FAL-PDU_req { dmpm_service_name := “DMPM_Data_req”, arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := “CS_ReqPDU”, fal_data := user_data) }, IncrementCounter(OSCC)	OPEN
S10	OPEN	CS_req && Role = “Client” || “Peer” && GetCounterValue(OSCC) < maxOSCC && CIU > 0 => FAL-PDU_req { dmpm_service_name := “DMPM_Data_req”, arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := “CS_ReqPDU”, fal_data := user_data) }, StartTimer(TC) IncrementCounter(OSCC)	OPEN

S11	OPEN	CS_req
		&& Role = “Client” || “Peer”
		=>
		CS_cnf (-) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			OPEN

#	Current state	Event or condition => action	Next state

S12	OPEN	CS_req
		&& Role = “Client” || “Peer”
		&& GetCounterValue(OSCC) ≥ maxOSCC
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Number of parallel services exceeded"
		},
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := " Number of parallel services exceeded",
		dlsdu := "null"
		},
		
		StopTimer
		ResetCounters
			CLOSED

S13	OPEN	CS_req
		&& Role = “Server”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Unallowed service as server"
		},
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := "Unallowed service as server",
		dlsdu := "null"
		},
		
		StopTimer
		ResetCounters
			CLOSED

S14	OPEN	CS_rsp && Role = “Server” || “Peer” && CIU = 0 => FAL-PDU_req { dmpm_service_name := "DMPM_Data_req", arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := "CS_RspPDU", fal_data := user_data) }, DecrementCounter(OSCS)	OPEN
S15	OPEN	CS_rsp && Role = “Server” || “Peer” && CIU > 0 => FAL-PDU_req { dmpm_service_name := "DMPM_Data_req", arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := "CS_RspPDU", fal_data := user_data) }, StartTimer(TS) DecrementCounter(OSCS)	OPEN

#	Current state	Event or condition => action	Next state

S17	OPEN	CS_rsp
		&& Role = “Client” II "Peer"
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Unallowed service as client"
		},
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := " Unallowed service as client",
		dlsdu := "null"
		},
		
		StopTimer
		ResetCounters
			CLOSED

S18	OPEN	UCA_req && Role = "Client" II "Peer" && GetCounterValue(UCC) < maxUCC && CIU = 0 => FAL-PDU_req { dmpm_service_name := "DMPM_Data_req", arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := "UCA_PDU", fal_data := user_data) }, IncrementCounter(UCC)	OPEN
S19	OPEN	UCA_req && Role = "Client" II "Peer" && GetCounterValue(UCC) < maxUCC && CIU > 0 => FAL-PDU_req { dmpm_service_name := "DMPM_Data_req", arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := "UCA_PDU", fal_data := user_data) }, StartTimer(TC) IncrementCounter(UCC)	OPEN

S20	OPEN	UCA_req
		&& Role = "Server" II "Peer"
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Unallowed service as server"
		},
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := "Unallowed service as server",
		dlsdu := "null"
		},
		
		StopTimer
		ResetCounters
			CLOSED

#	Current state	Event or condition => action	Next state

S22	OPEN	TCTimer expired
		=>
		StartTimer(TC)
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "IdlePDU",
		fal_data := "null")
		}
			OPEN

S23	OPEN	TSTimer expired
		=>
		StartTimer(TS)
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "IdlePDU",
		fal_data := "null")
		}
			OPEN

S24	NOT CLOSED	Abort_req => FAL-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepId (), reason := "disconnection–normal condition", dlsdu := BuildFAL-PDU (fal_pdu_name := "Abort_PDU", fal_id := identifier, fal_reason_code := reason_code, fal_additional_detail := additional_detail) }, StopTimer, ResetCounters	CLOSED

Table 31 – QUB-FC ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	CLOSED	Connect_ind
		&& Initiator = “True”
		=>
		FAL-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”
		&& FAL_Pdu_Type (dls_user_data) <> “EST_ReqPDU”
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := “Invalid FAL-PDU”,
		dlsdu := “null”
		}
			CLOSED

R3	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ReqPDU”
		&& AREPContextCheck (dlsdu) = "False"
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Context Check Negative",
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "Abort_PDU",
		fal_id := identifier,
		fal_reason_code := "Context Check Negative",
		fal_additional_detail := maxOSCC, maxOSCS, maxUCSC, maxUCSS, CI)
		}
			CLOSED

R4	CLOSED	unallowed primitive => (no actions taken)	CLOSED

R5	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ReqPDU”
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAL-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

R6	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ReqPDU”
		&& AREPContextCheck (dlsdu) = "True"
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlcepAddress = calling_dlcep_address
		=>
		MaxDlsduSizeFromRequestorNegotiated := dlsdu_size_from_requestor,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		
		EST_ind {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			RSP

#	Current state	Event or condition => action	Next state

R7	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ReqPDU”
		&& AREPContextCheck (dlsdu) = "True"
		&& RemoteAddressConfigurationType = “Free”
		=>
		RemoteDlcepAddress := calling_dlcep_address,
		MaxDlsduSizeFromRequestorNegotiated := dlsdu_size_from_requestor,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		
		EST_ind {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			RSP

R8	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAL-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

R9	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ReqPDU”
		&& RemoteAddressConfigurationType = “Free”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAL-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

R10	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) <> “EST_ReqPDU”
		&& RemoteAddressConfigurationType = “Free”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		}
			SAME

R11	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = "EST_ReqPDU"
		&& RemoteDlcepAddress = calling_dlcep_address
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		dlcep_dl_id := dlcep_dl_id (from Connect_ind),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Invalid FAL-PDU"
		}
			CLOSED

#	Current state	Event or condition => action	Next state

R12	REQ
	OPEN
		Connect_ind
		&& Initiator = “True”
		&& RemoteDlcepAddress <> calling_dlcep_address
		=>
		FAL-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

R13	REQ
	OPEN
		Connect_ind
		&& Initiator = “True”
		&& RemoteDlcepAddress = calling_dlcep_address
		=>
		FAL-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 := "FAL",
		reason_code := "Multiple Initiators"
		}
			CLOSED

R14	REQ	Connect_cnf
		&& FAL_Pdu_Type (dls_user_data) = “EST_RspPDU”
		=>
		MaxDlsduSizeFromRequestorNegotiated := dlsdu_size_from_requestor,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		DllPriorityNegotiated := dll_priority,
		
		EST_cnf(+) {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			OPEN

R15	REQ	Connect_cnf
		&& FAL_Pdu_Type (dls_user_data) <> “EST_RspPDU”
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Invalid FAL-PDU"
		}
			CLOSED

R16	REPL	FAL-PDU_ind && dmpm_service_name = “DMPM_Connection_Established_ind” &&CIU=0 => (no actions taken)	OPEN
R17	REPL	FAL-PDU_ind && dmpm_service_name = “DMPM_Connection_Established_ind” &&CIU>0 => StartTimer(RS)	OPEN

#	Current state	Event or condition => action	Next state

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

R19	REQ	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ErrPDU”
		=>
		EST_cnf(-) {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			CLOSED

R20	REQ	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fal_pdu <> “null”
		&& ((FAL_Pdu_Type (dls_user_data) <> “Abort_PDU”)
		&& (FAL_Pdu_Type (dls_user_data) <> “EST_ErrPDU”))
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid FAL-PDU”,
		additional_detail := “null”
		}
			CLOSED

R21	REQ
	RSP
		FAL-PDU_ind
		&& dmpm_service_name <> “DMPM_Disconnect_ind”
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid DL-Event",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := "Invalid DL-Event"
		}
			CLOSED

R22	REPL RSP	FAL-PDU_ind && dmpm_service_name = “DMPM_Disconnect_ind” && fal_pdu <> “null” && FAL_Pdu_Type (dls_user_data) <> “Abort_PDU” => Abort_ind{ arep_id := GetArepId (), locally_generated := "True", identifier := “FAL”, reason_code := “Invalid FAL-PDU”, additional_detail := “null” }, StopTimer, ResetCounters	CLOSED

#	Current state	Event or condition => action	Next state
R23	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Disconnect_ind” && fal_pdu <> “null” && FAL_Pdu_Type (dls_user_data) <> “Abort_PDU” && Role = "Client" II "Peer" => Abort_ind{ arep_id := GetArepId (), locally_generated := "True", identifier := “FAL”, reason_code := “Invalid FAL-PDU”, additional_detail := “null” }, StopTimer, ResetCounters	CLOSED
R24	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Disconnect_ind” && fal_pdu <> “null” && FAL_Pdu_Type (dls_user_data) <> “Abort_PDU” && Role = "Server" II "Peer" => Abort_ind{ arep_id := GetArepId (), locally_generated := "True", identifier := “FAL”, reason_code := “Invalid FAL-PDU”, additional_detail := “null” }, StopTimer, ResetCounters	CLOSED

R25	OPEN	FAL-PDU_ind
		&& ((dmpm_service_name <> “DMPM_Disconnect_ind”)
		II (dmpm_service_name <> “DMPM_Data_ind”))
		&& Role = "Client" II "Peer"
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid DL-Event",
		dlsdu := “null”
		},
		
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid DL-Event”,
		additional_detail := “null”
		},
		
		StopTimer,
		ResetCounters
			CLOSED

#	Current state	Event or condition => action	Next state

R26	OPEN	FAL-PDU_ind
		&& ((dmpm_service_name <> “DMPM_Disconnect_ind”)
		II (dmpm_service_name <> “DMPM_Data_ind”))
		&& Role = "Server" II "Peer"
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid DL-Event",
		dlsdu := “null”
		},
		
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid DL-Event”,
		additional_detail := “null”
		},
		
		StopTimer,
		ResetCounters
			CLOSED

R27	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && Role = “Peer” || “Server” && FAL_Pdu_Type (dls_user_data) = “CS_ReqPDU” && GetCounterValue(OSCS) < maxOSCS && CIU = 0 => CS_ind { arep_id := GetArepId (), user_data := fal_pdu }, IncrementCounter(OSCS)	OPEN
R28	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && Role = “Peer” || “Server” && FAL_Pdu_Type (dls_user_data) = “CS_ReqPDU” && GetCounterValue(OSCS) < maxOSCS && CIU > 0 => CS_ind { arep_id := GetArepId (), user_data := fal_pdu }, IncrementCounter(OSCS) , StartTimer(RS)	OPEN

R30	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Server” ||“Peer”
		&& FAL_Pdu_Type (dls_user_data) = “CS_ReqPDU”
		&& GetCounterValue(OSCS) ≥ maxOSCS
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Number of parallel services exceeded",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Number of parallel services exceeded”,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

#	Current state	Event or condition => action	Next state
R31	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && Role = “Client” || “Peer” && FAL_Pdu_Type (dls_user_data) = “CS_RspPDU” && CIU = 0 => CS_cnf { arep_id := GetArepId (), user_data := fal_pdu }, DecrementCounter(OSCC)	OPEN
R32	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && Role = “Client” || “Peer” && FAL_Pdu_Type (dls_user_data) = “CS_RspPDU” && CIU > 0 => CS_cnf { arep_id := GetArepId (), user_data := fal_pdu }, DecrementCounter(OSCC) , StartTimer(RC)	OPEN

R33	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& FAL_Pdu_Type (dls_user_data) = “UCA_ReqPDU”
		&& Role = "Server" II "Peer"
		&& CIU = 0
		=>
		UCA_ind {
		arep_id := GetArepId (),
		user_data := fal_pdu
		},
		
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Data_req”,
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "UCA_AckPDU),
		fal_data := "null")
		}
			OPEN

R34	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& FAL_Pdu_Type (dls_user_data) = “UCA_ReqPDU”
		&& Role = "Server" II "Peer"
		&& CIU > 0
		=>
		UCA_ind {
		arep_id := GetArepId (),
		user_data := fal_pdu
		},
		
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Data_req”,
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "UCA_AckPDU),
		fal_data := "null")
		},
		
		StartTimer(RS)
			OPEN

#	Current state	Event or condition => action	Next state

R35	OPEN	FAL-PDU_ind
		&& dmpm_service_name = DMPM_Data_ind"
		&& FAL_Pdu_Type (dls_user_data) = “UCA_ReqPDU”
		&& Role = "Client" II "Peer"
		=>
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := "Invalid Event for Role",
		dlsdu := "null"
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid Event for Role”,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

R36	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && FAL_Pdu_Type (dls_user_data) = “UCA_AckPDU” && Role = "Client" II "Peer" && GetCounterValue(UCC) > 0 && CIU = 0 => DecrementCounter(UCC)	OPEN
R37	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && FAL_Pdu_Type (dls_user_data) = “UCA_AckPDU” && Role = "Client" II "Peer" && GetCounterValue(UCC) > 0 && CIU > 0 => DecrementCounter(UCC), StartTimer(RC)	OPEN

R38	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& FAL_Pdu_Type (dls_user_data) = “UCA_AckPDU”
		&& Role = "Client" II "Peer"
		&& GetCounterValue(UCC) = 0
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "UCA_AckPDU received and UCC=0",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “ UCA_AckPDU received and UCC=0”,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

R39	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && FAL_Pdu_Type (dls_user_data) = “IdlePDU” && CIU > 0 && Role = "Client" II "Peer" => StartTimer(RC)	OPEN

#	Current state	Event or condition => action	Next state
R40	OPEN	FAL-PDU_ind && dmpm_service_name = “DMPM_Data_ind” && FAL_Pdu_Type (dls_user_data) = “IdlePDU” && CIU > 0 && Role = "Server" II "Peer" => StartTimer(RS)	OPEN

R41	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& FAL_Pdu_Type (dls_user_data) = “IdlePDU”
		&& CIU = 0
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid FAL-PDU”,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

R42	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = "Client"
		&& ((FAL_Pdu_Type (dls_user_data) <> “CS_RspPDU”)
		II (FAL_Pdu_Type (dls_user_data) <> "UCA_AckPDU)
		II (FAL_Pdu_Type (dls_user_data) <> "IdlePDU"))
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid FAL-PDU”,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

#	Current state	Event or condition => action	Next state

R43	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = "Server"
		&& ((FAL_Pdu_Type (dls_user_data) <> “CS_ReqPDU”)
		II (FAL_Pdu_Type (dls_user_data) <> “UCA_ReqPDU”)
		II (FAL_Pdu_Type (dls_user_data) <> "IdlePDU"))
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid FAL-PDU”,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

R44	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = "Client"
		&& ((FAL_Pdu_Type (dls_user_data) <> “CS_ReqPDU”)
		II (FAL_Pdu_Type (dls_user_data) <> “CS_RspPDU”)
		II (FAL_Pdu_Type (dls_user_data) <> “UCA_ReqPDU”)
		II (FAL_Pdu_Type (dls_user_data) <> "UCA_AckPDU)
		II (FAL_Pdu_Type (dls_user_data) <> "IdlePDU"))
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := “Invalid FAL-PDU”,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

R45	OPEN	RCTimer expired
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "RCTimer expired",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := "RCTimer expired",
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

#	Current state	Event or condition => action	Next state

R46	OPEN	RSTimer expired
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := "RSTimer expired",
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := "RSTimer expired",
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

R47	NOT CLOSED	FAL-PDU_ind && dmpm_service_name = “DMPM_Disconnect_ind” && fal_pdu <> “null” && FAL_Pdu_Type (dls_user_data) = “Abort_PDU” => Abort_ind{ arep_id := GetArepId (), locally_generated := "False", identifier := AbortIdentifier (fal_pdu), reason_code := AbortReason (fal_pdu), additional_detail := AbortDetail (fal_pdu) }, StopTimer, ResetCounters	CLOSED
R48	NOT CLOSED	FAL-PDU_ind && dmpm_service_name = “DMPM_Disconnect_ind” && fal_pdu = "null" && originator = "remote_dls_provider" => Abort_ind{ arep_id := GetArepId (), locally_generated := "False", identifier := “Data Link Layer”, reason_code := reason, additional_detail := "null" }, StopTimer, ResetCounters	CLOSED
R49	NOT CLOSED	FAL-PDU_ind && dmpm_service_name = “DMPM_Disconnect_ind” && fal_pdu = "null" && originator = "remote_dls_user" => Abort_ind{ arep_id := GetArepId (), locally_generated := "False", identifier := “FAL”, reason_code := reason, additional_detail := "null" }, StopTimer, ResetCounters	CLOSED

#	Current state	Event or condition => action	Next state
R50	NOT CLOSED	FAL-PDU_ind && dmpm_service_name = “DMPM_Disconnect_ind” && fal_pdu = "null" && originator = "local_dls_provider" => Abort_ind{ arep_id := GetArepId (), locally_generated := "True", identifier := “Data Link Layer”, reason_code := reason, additional_detail := "null" }, StopTimer, ResetCounters	CLOSED

R51	NOT CLOSED	ErrorToARPM
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := reason,
		dlsdu := “null”
		},
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := “FAL”,
		reason_code := reason,
		additional_detail := "null"
		},
		
		StopTimer,
		ResetCounters
			CLOSED

9.1.6.3 Functions used by QUB-FC ARPM

Table 32 through Table 44 define the functions used by this state machine.

Table 32 – Function GetArepId ()

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

Table 33 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
fal_pdu_name, calling_dlcep_address, called_dlcep_address, fal_data, fal_id, fal_reason_code, fal_additional_detail		dlsdu	
Function			
Builds an FAL-PDU out of the parameters given as input variables.			

Table 34 – Function FAL_Pdu_Type

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

Table 35 – Function AREPContextCheck()

Name	AREPContextCheck()	Used in	ARPM
Input		Output	
dl_sdu		Boolean value.	
Function			

This function checks the AREP context parameters that are received with establish service. The compatibility of the remote to the local context is shown in the following table:

Local Context		Remote Context
Type	=	Type
maxOSCC	≤	maxOSCS
maxOSCS	≤	maxUCSS
maxUCSC	≥	maxOSCC
maxUCSS	≥	maxUCSC
CI	=	CIU

Explanation:
≤: local value smaller than or equal to remote value
≥: local value larger than or equal to remote value
=: local value equal to remote value

Table 36 – Function AbortIdentifier

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

Table 37 – Function AbortReason

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

Table 38 – Function AbortDetail

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

NOTE The following two functions make use of persistent protocol timers that are able to issue the local events 'TSTimer expired', 'TCTimer expired', 'RCTimer expired' and 'RSTimer expired' to notify the expiration of the appropriate time interval to the ARPM.

Table 39 – Function StartTimer

Name	StartTimer	Used in	ARPM
Input identifier		Output	
Function	This function starts the selected persistent protocol timer as follows: If identifier is TS then function starts the TSTimer with the value of CIU/3. If identifier is TC then function starts the TCTimer with the value of CIU/3. If identifier is RS then function starts the RSTimer with the value of CIU. If identifier is RC then function starts the RCTimer with the value of CIU.		

NOTE The appropriate timer is restarted if it was still running.

Table 40 – Function StopTimer

Name	StopTimer	Used in	ARPM
Input		Output	
Function	This function stops all local persistent protocol timers of the ARPM.		

NOTE The following functions make use of local persistent variables OSCC (current value of outstanding services at client), UCC (current value of unconfirmed services at client) and OSCS (current value of outstanding services at server) that supports counting of outstanding services. The initial values of OSCC, UCC, OSCS are 0.

Table 41 – Function ResetCounters

Name	ResetCounters	Used in	ARPM
Input		Output	
Function	This function sets OSCC, UCC, OSCS to zero.		

Table 42 – Function IncrementCounter

Name	IncrementCounter	Used in	ARPM
Input identifier		Output	
Function	This function increments the selected counter.		

Table 43 – Function DecrementCounter

Name	DecrementCounter	Used in	ARPM
Input identifier		Output	
Function	This function decrements the selected counter.		

Table 44 – Function GetCounterValue

Name	GetCounterValue	Used in	ARPM
Input identifier		Output value	
Function	This function returns the current value of the selected counter.		

9.2 Buffered network-scheduled unidirectional (BNU) ARPM

9.2.1 Primitive definitions

9.2.1.1 Primitives exchanged between ARPM and FSPM

Table 45 and Table 46 list the primitives exchanged between the FSPM and the ARPM.

Table 45 – Primitives issued by FSPM to ARPM

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

Table 46 – Primitives issued by ARPM to FSPM

Primitive name	Source	Associated parameters	Functions
EST_cnf(+)	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish response(+) primitive from the ARPM to the FSPM.
EST_cnf(-)	ARPM	arep_id, user_data	This is an FAL internal primitive used to convey an Establish response(-) primitive from the ARPM to the FSPM.
Abort_ind	ARPM	arep_id, locally_generated, identifier, reason_code, additional_detail	This is an FAL internal primitive used to convey an Abort primitive from the ARPM to the FSPM.
UCS_ind	ARPM	arep_id, duplicate_fal_sdu, user_data, local_timeliness, remote_timeliness	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) indication primitive from the ARPM to the FSPM.

9.2.1.2 Parameters of FSPM/ARPM primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in Table 47.

Table 47 – Parameters used with primitives exchanged between FSPM and ARPM

Parameter name	Description
arep_id	This parameter is used to unambiguously identify an instance of the AREP that has issued a primitive. A means for such identification is not specified by this standard.
user_data	This parameter conveys FAL-User data.
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.

9.2.2 DLL mapping of BNU AREP class

9.2.2.1 BNU AREP class formal model

This subclause describes the mapping of the BNU AREP class 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 standard; rather, it defines how they are used by this AR class.

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

CLASS: **Bnu**
PARENT CLASS: **BufferedNetworkScheduledUnidirectionalAREP**
ATTRIBUTES:

1	(m)	KeyAttribute:	PublisherDlcepAddress
3	(m)	Attribute:	QosParameterSet
3.1	(m)	Attribute:	DlcepClass (Publisher, Subscriber)
3.7	(m)	Constraint:	DlcepClass = Publisher
3.7.2	(m)	Attribute:	MaxDlsduSizeFromRequestor
3.8	(m)	Constraint:	DlcepClass = Subscriber
3.8.2	(m)	Attribute:	MaxDlsduSizeFromResponder

DLL SERVICES:

1	(c)	Constraint:	DlcepClass (Publisher)
1.1	(m)	OpsService:	DL-Put
1.2	(o)	OpsService:	DL-Get
2	(c)	Constraint:	DlcepClass (Subsciber)
2.1	(o)	OpsService:	DL-Put
2.2	(m)	OpsService:	DL-Get
3	(m)	OpsService:	DL-Buffer-Received

9.2.2.2 Attributes

9.2.2.2.1 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 subattributes: 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 FAL.

9.2.2.2.2 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 either Publisher or Subscriber and corresponds to PUBLISHER and SUBSCRIBER defined by the DLL, respectively.

9.2.2.2.3 MaxDlsduSizeFromRequestor

This attribute is used if the Role attribute has a value of "PushPublisher" and specifies the maximum length of an FAL-PDU that can be sent from this AREP.

9.2.2.2.4 MaxDlsduSizeFromResponder

This attribute is used if the Role attribute has a value of "PushSubscriber" and specifies the maximum length of an FAL-PDU that can be received by this AREP.

9.2.2.3 DLL services

Refer to IEC 61158-3-8 for DLL service descriptions.

9.2.3 BNU AREP state machine

9.2.3.1 BNU ARPM states

The defined states together with their descriptions of the BNU ARPM are listed in Table 48 and Figure 27.

Table 48 – BNU ARPM states

State	Description
CLOSED	The AREP is defined, but not capable of sending or receiving FAL-PDUs.
REQUESTING	The AREP has issued an EST_req and waiting for an EST_cnf primitive.
OPEN	The AREP is defined and capable of sending or receiving FAL-PDUs.

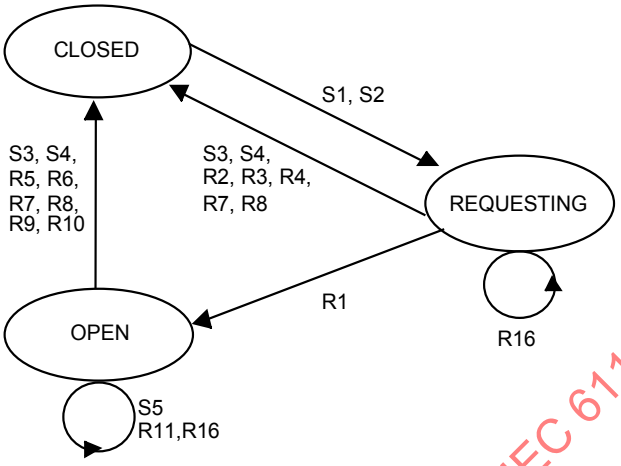


Figure 27 – State transition diagram of the BNU ARPM

9.2.3.2 BNU ARPM state table

Table 49 and Table 50 define the BNU ARPM state machine.

Table 49 – BNU ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1	CLOSED	EST_req
		&& Role = “PushPublisher”
		=>
		FAL-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	EST_req
		&& Role = “PushSubscriber”
		=>
		FAL-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

#	Current state	Event or condition => action	Next state

S3	NOT CLOSED	Abort_req
		&& Role = “PushPublisher”
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := “disconnection–normal condition”,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := “Abort_PDU”,
		fal_id := identifer,
		fal_reason_code := reason_code,
		fal_additional_detail := additional_detail)
		}
			CLOSED

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

S5	OPEN	UCS_req
		&& Role = “PushPublisher”
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Put_req”,
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := “UCS_PDU”,
		fal_data := user_data)
		}
			OPEN

Table 50 – BNU ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state

R1	REQ	Connect_cnf
		=>
		EST_cnf(+) {
		arep_id := GetArepId (),
		user_data := “null”
		}
			OPEN

R2	REQ	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fal_pdu = “null”
		&& originator = “local_dls_provider”
		=>
		EST_cnf(-) {
		arep_id := GetArepId (),
		user_data := “null”
		}
			CLOSED

R3	REQ	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fal_pdu = “null”
		&& originator = “remote_dls_provider”
		=>
		EST_cnf(-) {
		arep_id := GetArepId (),
		user_data := “null”
		}
			CLOSED

#	Current state	Event or condition => action	Next state

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

R5	OPEN	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fal_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

R6	OPEN	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fal_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

R7	NOT CLOSED	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& Fal_Pdu_Type (fal_pdu) = “Abort_PDU”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := “False”,
		identifier := AbortIdentifier (fal_pdu),
		reason_code := AbortReason (fal_pdu),
		additional_detail := AbortDetail (fal_pdu)
		}
			CLOSED

R8	NOT CLOSED	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fal_pdu <> “null”
		&& Fal_Pdu_Type (fal_pdu) <> “Abort_PDU”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := “True”,
		identifier := “FAL”,
		reason_code := “Invalid FAL-PDU”,
		additional_detail := “null”
		}
			CLOSED

#	Current state	Event or condition => action	Next state

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

R10	OPEN	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Buffer_Received_ind”
		&& FAL_pdu_type <> “UCS_PDU”
		=>
		Abort_ind{
		arep_id := GetArepId (),
		locally_generated := “True”,
		identifier := “FAL”,
		reason_code := “Invalid FAL-PDU”,
		additional_detail := “null”
		},
		
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		reason := “Invalid FAL-PDU”,
		dlsdu := “null”
		}
			CLOSED

R11	OPEN	FAL-PDU_ind
		&& Role = “PushSubscriber”
		&& dmpm_service_name = “DMPM_Buffer_Received_ind”
		&& FAL_Pdu_Type (fal_pdu) = “UCS_PDU”
		=>
		UCS_ind{
		arep_id := GetArepId (),
		duplicate_fal_sdu := duplicate_dlsdu,
		user_data := fal_pdu,
		local_timeliness := local_dle_timeliness,
		remote_timeliness := remote_dle_timeliness
		}
			OPEN

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

9.2.3.3 Functions used by BNU ARPM

Table 51 through Table 56 define the functions used by this state machine.

Table 51 – Function GetArepId ()

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

Table 52 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
fal_pdu_name, calling_dlcep_address, called_dlcep_address, fal_data, fal_id, fal_reason_code, fal_additional_detail		dlsdu	
Function			
Builds an FAL-PDU out of the parameters given as input variables.			

Table 53 – Function FAL_Pdu_Type

Name	FAL_Pdu_Type	Used in	ARPM
Input		Output	
fal_pdu		One of the FAL-PDU types defined in the clause 4.	
Function			
This function decodes the FAL-PDU that is conveyed in the fal_pdu parameter and retrieves one of the FAL-PDU types.			

Table 54 – Function AbortIdentifier

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

Table 55 – Function AbortReason

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

Table 56 – Function AbortDetail

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

9.3 Queued user-triggered bidirectional – transparent mode (QUB-TM) ARPM

9.3.1 QUB-TM Primitive definitions

9.3.1.1 Primitives exchanged between ARPM and FAL

The primitives exchanged between ARPM and FAL are shown in Table 57 and Table 58

Table 57 – Primitives issued by FAL to ARPM

Primitive name	Source	Associated parameters	Functions
DSA_req	FAL	user_data	This is a primitive used to convey a Data-Send-Acknowlegde request primitive from the FAL to the ARPM.

Table 58 – Primitives issued by ARPM to FAL

Primitive name	Source	Associated parameters	Functions
DSA_cnf(+)	ARPM	arep_id, user_data	This is a primitive used to convey a Data-Send-Acknowledge confirmation(+) primitive from the ARPM to the FAL.
DSA_cnf(-)	ARPM	arep_id, user_data	This is a primitive used to convey a Data-Send-Acknowledge confirmation(-) primitive from the ARPM to the FAL
DSA_ind	ARPM	arep_id, user_data	This is a primitive used to convey a Data-Send-Acknowledge indication from the ARPM to the FAL.

9.3.1.2 Parameters of FAL/ARPM primitives

The parameters used with the primitives exchanged between the FAL and the ARPM are described in Table 59.

Table 59 – Parameters used with primitives exchanged between FAL and ARPM

Parameter name	Description
Arep_id	This parameter is used to unambiguously identify an instance of the AREP that has issued a primitive. A means for such identification is not specified by this standard.
User_data	This parameter conveys FAL-User data.

9.3.2 DLL mapping of QUB-TM AREP Class

This subclause describes the mapping of the QUB-TM AREP class to the Type 8 Data Link Layer defined IEC 61158-3-8 and IEC 61158-4-8. It does not redefine the DLSAP attributes or DLME attributes that are or will be defined in the Data Link Layer standard; rather, it defines how they are used by this AR class.

NOTE A means to configure and monitor the values of these attributes are not in the scope of this standard.

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

CLASS: **Type 8 QubTM**
PARENT CLASS: **QueuedUser-TriggeredBidirectional-FlowControlAREP**
ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	QosParameterSet
3.3	(m)	Attribute:	MaxDlsduSizes
3.3.1	(m)	Attribute:	MaxDlsduSizeFromRequestor
3.3.2	(m)	Attribute:	MaxDlsduSizeFromResponder
3.3.3	(m)	Attribute:	MaxDlsduSizeFromRequestorNegotiated
3.3.4	(m)	Attribute:	MaxDlsduSizeFromResponderNegotiated
3.4	(m)	Attribute:	MaxQueueDepth
3.4.1	(m)	Attribute:	MaxSendingQueueDepth
3.4.2	(m)	Attribute:	MaximimReceivingQueueDepth

DLL SERVICES:

1	(m)	OpsService:	DL-Data

9.3.3 Attributes

9.3.3.1 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.

9.3.3.2 RemoteDlcepAddress

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

9.3.3.3 QosParameterSet

9.3.3.3.1 General

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.

9.3.3.3.2 MaxDlsduSize

MaxDlsduSizeFromRequestor

This attribute specifies the configured value of the maximum length of an FAL-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 requestor” parameter of the DLL.

MaxDlsduSizeFromResponder

This attribute specifies the configured value of the maximum length of an FAL-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.

9.3.3.3.3 MaxQueueDepth

MaxSendingQueueDepth

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

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

MaxReceivingQueueDepth

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

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

9.3.4 DLL services

Refer to IEC 61158-3-8 for DLL service descriptions.

9.3.5 QUB-TM AREP state machine

9.3.5.1 QUB-TM ARPM states

The defined states and their description of the QUB-TM are shown in Table 60 and Figure 28.

Table 60 – QUB-TM ARPM states

State name	Description
ACTIVE	The QUB-TM in the ACTIVE state is ready to transmit or receive primitives to or from the FAL and the ARPM.

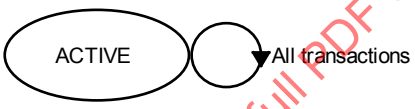


Figure 28 – State transition diagram of QUB-TM AREP

9.3.5.2 QUB-TM ARPM state table

The sender transactions are shown in Table 61 and the receiver transactions are shown in Table 62.

Table 61 – QUB-TM state table - sender transactions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	DSA_req
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Data_req”,
		arep_id := GetArepId(),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := “ DataSendAcknowledge-PDU”,
		fal_data := user_data;
		}
			ACTIVE

Table 62 – QUB-TM state table - receiver transactions

#	Current state	Event or condition => action	Next state

R1	ACTIVE	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& FAL_pdu_type(fal_pdu) = “ DataSendAcknowledge-PDU”
		=>
		DSA_ind {
		arep_id := GetArepID(),
		user_data = fal_pdu;
		}
			ACTIVE

R2	ACTIVE	DL_Data.cnf
		&& dl_status = “success”
		=>
		DSA_cnf(+) {
		arep_id := GetArepID(),
		}
			ACTIVE

R3	ACTIVE	DL_Data.cnf
		&& dl_status <> “success”
		=>
		DSA_cnf(-) {
		arep_id := GetArepID(),
		reason := dl_status
		}
			ACTIVE

NOTE In state S2 and S3 the Data Link Layer confirmation is used to build the DSA_cnf. The DataSendAcknowledge Service has no FAL response PDU.			

9.3.5.3 Functions used by QUB-TM ARPM

The functions used by the QUB-TM ARPM are shown in Table 63 through Table 69.

Table 63 – Function GetArepId ()

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

Table 64 – Function BuildFAL-PDU

Name	BuildFAL-PDU	Used in	ARPM
Input		Output	
fal_pdu_name, calling_dlcep_address, called_dlcep_address, fal_data, fal_id,		dlsdu	
Function			
Builds an FAL-PDU out of the parameters given as input variables.			

Table 65 – Function FAL_Pdu_Type

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

Table 66 – Function ResetCounters

Name	ResetCounters	Used in	ARPM
Input		Output	
Function			
This function sets OSCC, UCC, OSCS to zero.			

Table 67 – Function IncrementCounter

Name	IncrementCounter	Used in	ARPM
Input		Output	
identifier			
Function			
This function increments the selected counter.			

Table 68 – Function DecrementCounter

Name	DecrementCounter	Used in	ARPM
Input		Output	
identifier			
Function			
This function decrements the selected counter.			

Table 69 – Function GetCounterValue

Name	GetCounterValue	Used in	ARPM
Input		Output	
identifier		value	
Function			
This function returns the current value of the selected counter.			

10 DLL mapping protocol machine

10.1 Overview

The DLL Mapping Protocol Machine is common to all the AREP types. Only applicable primitives are different among different AREP types.

Remarks about dl identifiers:

The Data Link Layer (see IEC 61158-3-8) defines two types of identifiers to distinguish each DL-primitive or to match one DL outgoing primitive with its mate incoming primitive. They are suffixed as dl-identifier or DLS-user-identifier. In a real implementation of an FAL-DL interface, these identifications may be achieved by means of a pointer to memory location or a return value of a function call, or something else. Since they are purely a local matter, it is not testable over the Fieldbus network. For this reason, they are not included as parameters of the DMPM primitives.

“dl-identifiers” or “dls-user-identifiers“are mandatory in the DL-services. The FAL assumes that the values of these parameters are provided from DLSAPs or DLCEPs by a local means.

A remark about DLS-user identification:

It is assumed that a connection between one ARPM instance and one DMPM instance is established locally other than a protocol means. Therefore, DLS-user identification parameters are not used in the DMPM primitives.

A remark about buffer or queue identifiers:

The Data Link Layer standard uses parameters to identify a buffer that are shared between the Data Link Layer and the DLS-user. Although they are useful to clarify the operations of the Data Link Layer, none of them affects protocol behavior of the FAL and DL. In a real implementation, these parameters are implementation dependent. Therefore, this standard does not include parameters that directly correspond to these buffer identifiers. A means for identifying the buffers between the FAL and the DL is a local matter.

A remark about initialization of the Data Link Layer:

The FAL assumes that the initialization procedures of the Data Link Layer have been executed prior to the operations of the FAL state machines. Therefore, DL-services to deal with such initialization procedures are not listed in IEC 61158-3.

A remark about DLC establishment:

The FAL assumes that all DLCs have been established prior to the operation of the FAL state machine. Therefore, DL-services to deal with such establishment procedures are not listed.

10.2 Primitive definitions

10.2.1 Primitives exchanged between DMPM and ARPM

Table 70 shows the primitives issued by ARPM to DMPM.

Table 70 – Primitives issued by ARPM to DMPM

Primitive names	Source	Associated parameters	Functions
FAL-PDU_req	ARPM	dmpm_service_name, arep_id, called_address*), calling_address*), responding_address*), local_dlcep_address*), reason, action_class*), dlsdu	This primitive is used to request the DMPM to transfer an FAL-PDU, or to request an abort without transferring an FAL-PDU. It passes the FAL-PDU to the DMPM as a DLSDU. It also carries some of the Data Link Layer parameters that are referenced there.
NOTE These parameters are issued by the ARPM state machines but they are not used by the Data Link Layer.			

Table 71 shows the primitives issued by DMPM to ARPM.

Table 71 – Primitives issued by DMPM to ARPM

Primitive names	Source	Associated parameters	Functions
FAL-PDU_ind	DMPM	dmpm_service_name, originator, reason, duplicate_dlsdu, calling_address, fal_pdu, local_dle_timeliness, remote_dle_timeliness	This primitive is used to pass an FAL-PDU received as a Data Link Layer service data unit to a designated ARPM. It also carries some of the Data Link Layer parameters that are referenced in the ARPM.
Connect_ind	DMPM	calling_dlcep_address, dlsdu_size_from_requestor, 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	dll_priority, dlsdu_size_from_requestor, 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.
ErrorToARPM	DMPM	originator, reason	This primitive is used to convey selected communication errors reported by the Type8 Data Link Layer to a designated ARPM.

10.2.2 Parameters of ARPM/DMPM primitives

The parameters used with the primitives exchanged between the ARPM and the DMPM are described in Table 72.

Table 72 – Parameters used with primitives exchanged between ARPM and DMPM

Parameter name	Description
arep_id	This parameter carries a local identifier to specify the associated AR instance.
action_class	This parameter conveys the value of the dl_action_class parameter.
calling_address	This parameter conveys the value of the dl_calling_address.
calling_dlcep_address	This parameter conveys the value of the RequestingAREP parameter supplied with the EST_ReqPDU.
called_address	This parameter conveys the value of the dl_called_address parameter.
dmpm_service_name	This parameter conveys a Type8 Data Link Layer service name. Possible values are all the DL-X.yyy primitives defined in this section and are represented as DMPM_X_yyy.
local_dlcep_address	This parameter conveys the value of the dl_dlcep_address parameter.
duplicate_dlsdu	As duplicate_dlsdu detection is not supported by the Data Link Layer this attribute is always set to "False".
dll_priority	As no different priorities are supported by the Data Link Layer this attribute is always set to "normal".
dlsdu_size_from_requestor	This parameter conveys the value of the dl_dlsdu_size_from_requestor parameter.
dlsdu_size_from_responder	This parameter conveys the value of the dl_dlsdu_size_from_responder parameter.
dls_user_data	This parameter conveys the value of the dl_dls_user_data parameter.
dlsdu	This parameter conveys the value of the dl_dls_user_data parameter.
fal_pdu	This parameter conveys the value of the dl_dls_user_data parameter.
remote_dle_timeliness	As timeliness is not supported by the Data Link Layer this attribute is always set to "False".
local_dle_timeliness	As timeliness is not supported by the Data Link Layer this attribute is always set to "False".
originator	This parameter conveys the value of the dl_originator parameter.
reason	This parameter conveys the value of the dl_reason parameter.
responding_address	This parameter conveys the value of the dl_responding_address parameter.

10.2.3 Primitives exchanged between data-link layer and DMPM

NOTE The DLL primitives (see Table 73) and their parameters are defined in IEC 61158-3-8.

Table 73 – Primitives exchanged between data-link layer and DMPM

Primitive names	Source	Associated parameters
DL_Data.ind	Data Link Layer	dl_dls_user_data
DL_Data.cnf	Data Link Layer	dl_status
DL_Buffer_Received.ind	Data Link Layer	dl_status
DL_Get.cnf(+)	Data Link Layer	dl_dls_user_data
DL_Get.cnf(-)	Data Link Layer	dl_error_type
DL_Put.cnf(+)	Data Link Layer	(none)
DL_Put.cnf(-)	Data Link Layer	dl_error_type
DL_Data.req	DMPM	dl_dls_user_data
DL_Get.req	DMPM	(none)
DL_Put.req	DMPM	dl_dls_user_data

10.2.4 Parameters of DMPM/data-link layer primitives

The parameters used with the primitives exchanged between the DMPM and the data-link layer are defined in the DLL Service definition (see IEC 61158-3-8). They are prefixed by "dl_" to indicate that they are used by the FAL.

10.3 DMPM state machine

10.3.1 DMPM states

The defined state of the DMPM together with its description listed in Table 74 and Figure 29 shows the corresponding state transition diagram.

Table 74 – 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 Link Layer and the ARPM.

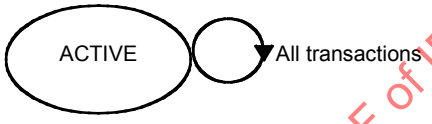


Figure 29 – State transition diagram of DMPM

10.3.2 DMPM state table

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

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

Table 75 shows the sender transaction state table:

Table 75 – DMPM state table – sender transactions

#	Current state	Event or condition => action	Next state

S1a1	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = "Unlinked"
		&& (DMPM_service_name = "DMPM_Connect_req"
		|| DMPM_service_name = "DMPM_Connect_rsp"
		|| DMPM_service_name = "DMPM_Data_req"
		|| DMPM_service_name = "DMPM_Put_req")
		=>
		PickArep (arep_id),
		
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Disconnect_ind",
		originator := local_dls_provider,
		reason := "Unlinked",
		fal_pdu := "null"
		}
			ACTIVE

S1a2	ACTIVE	FAL-PDU_req && DlLinkStatus() = "Unlinked" && DMPM_service_name = "DMPM_Disconnect_req" && (ArepRole(arep_id) = "Client" || ArepRole(arep_id) = "Server" || ArepRole(arep_id) = "Peer") => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state

S1b1	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Connect_req"
		&& (ArepRole(arep_id) = “Client”
		|| ArepRole(arep_id) = “Peer”)
		=>
		PickArep (arep_id),
		
		DL_Data.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S1b2	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Connect_req"
		&& ArepRole(arep_id) = “PushPublisher”
		=>
		PickArep (arep_id),
		
		Connect_cnf {
		dlsdu_size_from_requestor := MaxDlsduSizeFromRequestor,
		dlsdu_size_from_responder := “null”,
		dls_user_data := “null”
		}
			ACTIVE

S1b3	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Connect_req"
		&& ArepRole(arep_id) = “PushSubscriber”
		=>
		PickArep (arep_id),
		
		Connect_cnf {
		dlsdu_size_from_requestor := "null",
		dlsdu_size_from_responder := MaxDlsduSizeFromResponder,
		dls_user_data := “null”
		}
			ACTIVE

S2	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Connect_rsp"
		=>
		PickArep (arep_id),
		
		DL_Data.req {
		dl_dls_user_data := dlsdu
		}
		FAL-PDU_ind {
		dmpm_service_name := "DMPM_Connection_Established_ind",
		fal_pdu := "null"
		}
			ACTIVE

S3a	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Disconnect_req"
		&& (ArepRole(arep_id) = “Client”
		|| ArepRole(arep_id) = “Server”
		|| ArepRole(arep_id) = “Peer”
		=>
		PickArep (arep_id),
		
		DL_Data.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S3b	ACTIVE	FAL-PDU_req && DMPM_service_name = "DMPM_Disconnect_req" && (ArepRole(arep_id) = “PushPublisher” || ArepRole(arep_id) = “PushSubscriber”) => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state

S4	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Data_req"
		=>
		PickArep (arep_id),
		
		DL_Data.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S6a	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Put_req"
		&& ArepRole(arep_id) <> "PushPublisher"
		=>
		PickArep (arep_id),
		
		DL_Put.req {
		dl_dls_user_data := dlsdu
		}
		
		DL_Put.cnf(+) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := “DMPM_Put_cnf”,
		reason := “success”
		}
		
		DL_Put.cnf(-) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := “DMPM_Put_cnf”,
		reason := dl_error_type
		}
		
		ErrorToARPM {
		originator := “local_dls_provider”,
		reason := dl_error_type
		}
			ACTIVE

S6b	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = "DMPM_Put_req"
		&& ArepRole(arep_id) = "PushPublisher"
		=>
		PickArep (arep_id),
		
		DL_Put.req {
		dl_dls_user_data := RemoveUcsPduHeader(dlsdu)
		}
		
		DL_Put.cnf(+) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := “DMPM_Put_cnf”,
		reason := “success”
		}
		
		DL_Put.cnf(-) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := “DMPM_Put_cnf”,
		reason := dl_error_type
		}
		
		ErrorToARPM {
		originator := “local_dls_provider”,
		reason := dl_error_type
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

S7a	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = “DMPM_Get_req”
		&& ArepRole(arep_id) <> "PushSubscriber"
		=>
		PickArep (arep_id),
		
		DL_Get.req {
		}
		
		DL_Get.cnf(+) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Get_cnf",
		reason :=”success”,
		fal_pdu := dl_dls_user_data
		}
		
		DL_Get.cnf(-) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Get_cnf",
		reason :=dl_error_type
		}
			ACTIVE

S7b	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = “DMPM_Get_req”
		&& ArepRole(arep_id) = "PushSubscriber"
		=>
		PickArep (arep_id),
		
		DL_Get.req {
		}
		
		DL_Get.cnf(+) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Get_cnf",
		reason :=”success”,
		fal_pdu := AddUcsPduHeader(dl_dls_user_data)
		}
		
		DL_Get.cnf(-) -- This is an immediate response.
		=>
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Get_cnf",
		reason :=dl_error_type
		}
			ACTIVE

S8a	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Unlinked”
		&& DMPM_service_name = “DMPM_Compel_req”
		=>
		PickArep (arep_id),
		
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Compel_Service_cnf",
		reason := "failure-inappropriate request”
		}
			ACTIVE

S8b	ACTIVE	FAL-PDU_req
		&& DlLinkStatus() = “Linked”
		&& DMPM_service_name = “DMPM_Compel_req”
		=>
		PickArep (arep_id),
		
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Compel_Service_cnf",
		reason := "failure-inappropriate request”
		}
			ACTIVE

Table 76 shows the receiver transaction state table:

Table 76 – DMPM state table – receiver transactions

#	Current state	Event or condition => action	Next State
R15a	ACTIVE	DL_Data.ind && FindArep() <> "True" => (no actions taken)	ACTIVE

R16a	ACTIVE	DL_Data.ind
		&& FindArep() = "True"
		&& FalArHeader (dl_dls_user_data) = “Establish-Request-PDU”
		=>
		Connect_ind {
		calling_dlcep_address :=RemoteDlcepAddress,
		dlsdu_size_from_requestor := MaxDlsduSizeFromRequestor,
		dlsdu_size_from_responder := MaxDlsduSizeFromResponder,
		dls_user_data := dl_dls_user_data
		}
			ACTIVE

R16b	ACTIVE	DL_Data.ind
		&& FindArep() = "True"
		&& FalArHeader (dl_dls_user_data) = “Establish-Response-PDU”
		=>
		Connect_cnf {
		dll_priority := "normal",
		dlsdu_size_from_requestor := MaxDlsduSizeFromRequestor,
		dlsdu_size_from_responder := MaxDlsduSizeFromResponder,
		dls_user_data := dl_dls_user_data
		}
			ACTIVE

R16c	ACTIVE	DL_Data.ind
		&& FindArep() = "True"
		&& (FalArHeader (dl_dls_user_data) = “Establish-Error-PDU”
		|| FalArHeader(dl_dls_user_data) = ”Abort-PDU”)
		=>
		FAL-PDU_ind{
		DMPM_service_name = ”DMPM_Disconnect_ind”,
		fal_pdu := LSDU
		}
			ACTIVE

R16d	ACTIVE	DL_Data.ind
		&& FindArep() = "True"
		&& FalArHeader(dl_dls_user_data) <> “Establish-Request-PDU”
		&& FalArHeader(dl_dls_user_data) <> “Establish-Response-PDU”
		&& FalArHeader(dl_dls_user_data) <> “Establish-Error-PDU”
		&& FalArHeader(dl_dls_user_data) <> “Abort-PDU”
		=>
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Data_ind",
		fal_pdu := LSDU
		}
			ACTIVE

R18	ACTIVE	DL_Data.cnf && dl_status = “OK” => (no actions taken)	ACTIVE
R19a	ACTIVE	DL_Data.cnf && dl_status <> “OK” && FindArep() = "False" => (no actions taken)	ACTIVE

R19b	ACTIVE	DL_Data.cnf
		&& dl_status <> “OK”
		&& FindArep() = "True"
		=>
		FAL-PDU_ind {
		DMPM_service_name := “DMPM_Data_cnf”,
		reason := dl_status
		}
		ErrorToARPM {
		originator := “local_dls_provider”,
		reason := dl_status
		}
			ACTIVE

#	Current state	Event or condition => action	Next State

R21	ACTIVE	DL_Buffer_Received.ind
		=>
		--loop through all AREPs
		
		arep_id := SelectNextArep()
		
		if (ArepRole(arep_id) = “PushSubscriber”)
		DL_Get.req {
		}
		
		DL_Get.cnf(+)
		
		FAL-PDU_ind {
		DMPM_service_name := "DMPM_Buffer_Received_ind",
		duplicate_dlsdu := "False",
		fal_pdu := AddUcsPduHeader(dl_dls_user_data),
		local_dle_timeliness := "False",
		remote_dle_timeliness := "False"
		}
		
		DL_Get.cnf(-)
		
		FAL-PDU_ind {
		DMPM_service_name := “DMPM_Get_cnf”,
		reason := dl_error_type
		}
		ErrorToARPM {
		originator := “local_dls_provider”,
		reason := dl_error_type
		}
		endif
		
		if (ArepRole(arep_id) = “PushPublisher”)
		FAL-PDU_ind {
		DMPM_service_name := “DMPM_Buffer_Sent_ind”
		}
		endif
			ACTIVE

10.3.3 Functions used by DMPM

Table 77 through Table 84 show the functions used by the DMPM

Table 77 – 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 78 – Function FindAREP

Name	FindAREP	Used in	DMPM
Input		Output	
(local mapping)		True || False	
Function			
This function identifies the AREP that shall be bound with an active DMPM. True means the AREP exists. If it does, this function also returns a means to send a DMPM primitive to that AREP.			

Table 79 – Function SelectNextArep

Name	SelectNextAREP	Used in	DMPM
Input		Output	
(none)		arep_id	
Function			
Returns the arep_id parameter of the next existing AREP, starting with the first one.			

Table 80 – Function ArepRole

Name	ArepRole	Used in	DMPM
Input		Output	
arep_id		(the value of the Role attribute of the specified AREP)	
Function			
This function returns the Role attribute of the AREP specified by the arep_id parameter.			

Table 81 – Function FalArHeader

Name	FalArHeader	Used in	DMPM
Input		Output	
dl_dls_user_data		(the FAL header)	
Function			
This function decodes the dlsdu conveyed by the dl_dls_user_data argument and returns the FAL-PDU type.			

Table 82 – Function AddUcsPduHeader

Name	AddUcsPduHeader	Used in	DMPM
Input		Output	
dl_dls_user_data		fal_pdu	
Function			
Adds the UCS-PDU header to the dl_dls_user_data received from the DLE			

Table 83 – Function RemoveUcsPduHeader

Name	FalArHeader	Used in	DMPM
Input		Output	
fal_pdu		dlsdu	
Function			
Removes the UCS-PDU header from the FAL-PDU received from an AREP			

Table 84 – Function DlLinkStatus

Name	DlLinkStatus	Used in	DMPM
Input		Output	
(none)		Linked || Unlinked	
Function			
This function returns whether or not the local Dls-provider is synchronized at the moment.			

Bibliography

IEC/TR 61158-1 (Ed.2.0), *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61784-1 (Ed.2.0), *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 9506-2, *Industrial automation systems – Manufacturing Message Specification – Part 2: Protocol specification*

ISO/IEC 10646 (all parts), *Information technology – Universal Multiple-Octet Coded Character Set (UCS)*

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

ISO 8649, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element*

ISO 8650 (all parts), *Information technology – Open Systems Interconnection – Connection-oriented protocol for the Association Control Service Element: Protocol specification*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DE BUS DE TERRAIN –

Partie 6-8: Spécification de protocole de couche application – Éléments de Type 8

AVANT-PROPOS

1) La CEI (Commission Électrotechnique Internationale) 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. À 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 des questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux 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. Toute divergence entre toute Publication de la CEI et toute publication nationale ou régionale correspondante doit être indiquée en termes clairs dans cette dernière.
5) La CEI n'a prévu aucune procédure de marquage valant indication d'approbation et n'engage pas sa responsabilité pour les équipements déclarés conformes à une de ses Publications.
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.

NOTE L'utilisation de certains des types de protocoles associés est limitée par les détenteurs de leurs droits de propriété intellectuelle. Dans tous les cas, l'engagement à un abandon limité des droits de propriété intellectuelle pris par les détenteurs de ces droits permet d'utiliser un type particulier de protocole de couche liaison de données avec des protocoles de couche physique et de couche application dans des combinaisons de types telles que spécifiées de façon explicite dans la série CEI 61784. L'utilisation des divers types de protocoles dans d'autres combinaisons peut exiger la permission donnée par les détenteurs respectifs de leurs droits de propriété intellectuelle.

La Commission Électrotechnique Internationale (CEI) attire l'attention sur le fait qu'il est déclaré que la conformité avec les dispositions du présent document peut impliquer l'utilisation d'un brevet comme suit, où la notation [xx] indique le détenteur du droit de propriété:

Type 8:

DE 197 39 297 C2	[PxC]	"Automatisierungssystem und Steuervorrichtung zur transparenten Kommunikation zwischen verschiedenen Netzwerken."
US 2002/0042845 A1	[PxC]	"Automation System and connecting Apparatus for the Transparent Communication between two Networks."

La CEI ne prend pas de position quant à la preuve, à la validité et à la portée de ces droits de propriété.

Les détenteurs de ces droits de propriété ont donné l'assurance à la CEI qu'ils consentent à négocier des licences avec des demandeurs du monde entier, à des termes et conditions raisonnables et non discriminatoires. À ce propos, la déclaration des détenteurs des droits de propriété est enregistrée à la CEI. Des informations peuvent être obtenues auprès de:

[PxC]: Phoenix Contact GmbH & Co. KG
Intellectual Property Licenses & Standards
Flachsmarktstr. 8
D-32825 Blomberg,
Allemagne

La Norme internationale CEI 61158-6-8 a été établie par le sous-comité 65C: Réseaux de communications industrielles, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette première édition et ses parties d'accompagnement de la sous-série CEI 61158-6 annulent et remplacent la CEI 61158-6:2003. La présente édition de cette partie constitue une révision technique.

Par rapport à l'édition précédente, la présente édition de la CEI 61158-6 comporte les modifications significatives suivantes:

a) suppression du précédent bus de terrain de Type 6 en raison du manque de pertinence commerciale;
b) ajout de nouveaux types de bus de terrain;
c) division de la partie 6 de la troisième édition en plusieurs parties numérotées -6-2, -6-3, ….

La présente version bilingue (2013-02) correspond à la version anglaise monolingue publiée en 2007-12.

Le texte anglais de cette norme est issu des documents 65C/476/FDIS et 65C/487/RVD.

Le rapport de vote 65C/487/RVD donne toute information sur le vote ayant abouti à l'approbation de cette norme.

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

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

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

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

NOTE La révision de la présente norme sera synchronisée avec les autres parties de la série CEI 61158.

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

INTRODUCTION

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

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

- comme un guide pour les développeurs et les concepteurs;
- pour utilisation en essai et approvisionnement d'équipements;
- comme une partie d'un accord pour l'admission des systèmes dans l'environnement à systèmes ouverts;
- comme un affinement à la compréhension des communications à temps critiques dans le modèle OSI.

La présente norme est concernée, en particulier, par la communication et l'interopérabilité des capteurs, effecteurs et autres dispositifs d'automation. En utilisant la présente norme avec d'autres normes positionnées dans le modèle l'OSI ou dans des modèles de référence de bus de terrain, d'autres systèmes incompatibles peuvent travailler ensemble dans toutes combinaisons.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DE BUS DE TERRAIN –

Partie 6-8: Spécification de protocole de couche application – Éléments de Type 8

1 Domaine d'application

1.1 Généralités

La couche application de bus de terrain (FAL «Fieldbus Application Layer») fournit aux programmes d'utilisateur un moyen d'accéder à l'environnement de communication du bus de terrain. À cet égard, la FAL peut être vue comme une «fenêtre entre des programmes d'application correspondants».

La présente norme fournit les éléments communs pour les communications de messagerie de base à temps critique et à temps non critique entre des programmes d'application dans un environnement d'automation et le matériau spécifique au bus de terrain de Type 8. Le terme "en temps critique" sert à représenter la présence d'une fenêtre temporelle, dans les limites de laquelle une ou plusieurs actions spécifiées sont tenues d'être parachevées avec un certain niveau 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, l'installation et éventuellement pour la vie humaine.

La présente norme spécifie les interactions entre les applications distantes et définit le comportement visible de l'extérieur fourni par la couche application de bus de terrain de Type 8 en termes de:

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

b) syntaxe de transfert définissant les règles de codage qui sont appliquées aux unités de données de protocole de couche application;

c) diagramme d'états de contexte application définissant le comportement de service application visible entre des entités d'application engagées dans une communication;

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

Le but de la présente norme est de définir le protocole fourni pour

1) définir la représentation câblée des primitives de service définies dans la CEI 61158-5-8, et

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

La présente norme spécifie le protocole de la couche application des bus de terrain de Type 8, en conformité avec le Modèle de référence de base de l'OSI (ISO/CEI 7498) et la structure de la couche application de l'OSI (ISO/CEI 9545).

1.2 Spécifications

L'objet principal de la présente norme est de spécifier la syntaxe et le comportement du protocole de couche application qui achemine les services de couche application définis dans la CEI 61158-5-8.

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles de communications industrielles préexistants. C'est ce dernier objectif qui donne naissance à la diversité des protocoles normalisés dans la série CEI 61158-6.

1.3 Conformité

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

2 Références normatives

Les documents de référence suivants sont indispensables pour l'application du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

CEI 60559, *Arithmétique binaire en virgule flottante pour systèmes à microprocesseurs*

CEI 61158-3-8, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-8: Définition des services des couches de liaison de données – Éléments de Type 8*

CEI 61158-4-8, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-8: Spécification de protocole de la couche liaison de données – Éléments de Type 8*

CEI 61158-5-8, *Industrial communication networks – Fieldbus specifications – Part 5-8: Application layer service definition – Type 8 elements* (disponible en anglais seulement)

ISO/CEI 7498 (toutes les parties), *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base*

ISO/CEI 8822, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Définition du service de présentation*

ISO/CEI 8824, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1):*

ISO/CEI 8825, *Technologies de l'information – Règles de codage ASN.1 Spécification des règles de codage de base (BER), des règles de codage canoniques (CER) et des règles de codage distinctives (DER)*

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

3 Termes et définitions

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

3.1 Termes de l'ISO/CEI 7498-1

La présente norme est basée en partie sur les concepts développés dans l'ISO/CEI 7498-1 et utilise les termes suivants définis ci-après:

3.2
application entity (entité d'application)

3.3
application process (processus d'application)

3.4
application protocol data unit (unité de donnée de protocole application)

3.5
application service element (élément de service application)

3.6
application entity invocation (invocation d'entité d'application)

3.7
application transaction (transaction d'application)

3.8
transfer syntax (syntaxe de transfert)

3.9 Termes de l'ISO/CEI 8822

Pour les besoins du présent document, le terme suivant tel que défini dans l'ISO/CEI 8822 s'applique:

3.10
abstract syntax (syntaxe abstraite)

3.11 Termes de l'ISO/CEI 9545

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

3.12
application-context (contexte d'application)

3.13
application-process-type (type de processus d'application)

3.14
application-service-element (élément de service application)

3.15
Application Control Service Element (élément de service de contrôle d'association)

3.16 Termes de l'ISO/CEI 8824

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 8824 s'appliquent:

3.17
type any (n'importe quel type)

3.18
type bitstring (chaîne binaire)

3.19
type Boolean (booléen)

3.20
type choice (choix)

3.21
type de composant

3.22
false (faux)

3.23
type integer ("entier")

3.24
module

3.25
type null ("vide")

3.26
identificateur d'objet

3.27
type octetstring ("chaîne d'octets")

3.28
production

3.29
type simple

3.30
type sequence of (séquence de)

3.31
type sequence (séquence)

3.32
type structuré

3.33
marqueur

3.34
type tagged ("marqué")

3.35
true (vrai)

3.36
type

3.37 Termes de l'ISO/CEI 8825

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 8825 s'appliquent:

3.38
codage (d'une valeur de donnée)

3.39
valeur de la donnée

3.40 Termes et définitions issus de la CEI 61158-5-8

3.41
relation d'applications

3.42
client

3.43
classe d'erreurs

3.44
publisher (éditeur)

3.45
server (serveur)

3.46
subscriber (abonné)

3.47 Autres termes et définitions

Les termes et définitions suivants sont utilisés dans la présente norme.

3.48
appelé
utilisateur de service ou fournisseur de service qui reçoit une primitive "indication" ou une APDU de demande

3.49
appelant
utilisateur de services ou fournisseur de services qui lance une primitive "indication" ou une APDU de demande

3.50
informations de gestion
informations accessibles du réseau qui prennent en charge la gestion de l'environnement du bus de terrain

3.51
récepteur
utilisateur de service qui reçoit une primitive confirmée ou une primitive non conformée, ou fournisseur de service qui reçoit une APDU confirmée ou une APDU non confirmée

3.52
ressource
capacité de traitement ou d'informations d'un sous-système

3.53
expéditeur
utilisateur de service qui envoie une primitive confirmée ou une primitive non confirmée, ou fournisseur de service qui envoie une APDU confirmée ou une APDU non confirmée

4 Description de la syntaxe de FAL

4.1 Syntaxe abstraite des AR PDU de la FAL

4.1.1 Définition de haut niveau

Les productions définies ici doivent être utilisées avec les règles de codage de périphériques (Peripherals Encoding Rules, voir 5.2) pour le codage des APDU.

```
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 (envoi confirmé)

```
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 (envoi non confirmé)

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

4.1.4 Service Unconfirmed acknowledge send (envoi d'acquittement non confirmé)

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

4.1.5 Service Establish (établir)

```
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
}
```

4.1.6 MaxOSCC

MaxOSCC ::= Unsigned8

4.1.7 MaxOSCS

MaxOSCS ::= Unsigned8

4.1.8 MaxUCSC

MaxUCC ::= Unsigned8

4.1.9 MaxUCSS

MaxUCS ::= Unsigned8

4.1.10 CIU

CIU ::= Unsigned32

4.1.11 InvokeID

InvokeID ::= Unsigned8

4.1.12 ConType

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

4.1.13 Service Data send acknowledge (acquittement d'envoi de données)

```
DataSendAcknowledge-PDU ::= SEQUENCE {
      Protocol-Code,[APPLICATION 8],Address2ARP,      --- Protocol-Code (Code de protocole) dans le quartet de
poids fort de l'octet!!!
      Block-Number,
      Block-Length,
      Protocol-Data,
}
```

4.1.14 Protocol-Code

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

4.1.15 Block-Number

```
Block-Number ::= Unsigned8                 --- 0  pas de groupage
                                           ---- 1…254 numéro de bloc
                                           ---- 255 dernier bloc
```

4.1.16 Block-Length

Block-Length ::= Unsigned

4.1.17 Protocol-Data

```
Protocol-Data::= Any                       --- transfert de protocole transparent
```

4.1.18 Address2 ARP

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

4.1.19 Destination-Address

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

4.1.20 Source-Address

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

4.1.21 Destination-Node

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

4.1.22 Source-Node

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

4.1.23 Destination-Subnode

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

4.1.24 Source-Subnode

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

4.2 Syntaxe abstraite de PDUBody

4.2.1 Service Abort (arrêter prématurément)

```
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 Error Type

```
ErrorType ::= SEQUENCE {
    errorClass                      [0] IMPLICIT ErrorClass,
    optionalParametersMap           [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    additionalCode                  [1] IMPLICIT Integer16 OPTIONAL,
    additionalDescription           [2] IMPLICIT VisibleString OPTIONAL,
}
```

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

```
ErrorClass ::=  CHOICE
{
    vfdState                [1]   IMPLICIT  Integer8 {
                                      other                           (0)
    },
    applicationReference    [2]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      application-unreachable         (1),
                                      application-reference-invalid   (2),
                                      context-unsupported             (3)
    },
    definition              [3]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      object-undefined                (1),
                                      object-attributes-inconsistent  (2),
                                      name-already-exists             (3),
                                      type-unsupported                (4),
                                      type-inconsistent               (5)
    },
    resource                [4]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      memory-unavailable              (1)
    },
    Service                 [5]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      object-state-conflict           (1),
                                      pdu-size                        (2),
                                      object-constraint-conflict      (3),
                                      parameter-inconsistent          (4),
                                      illegal-parameter               (5)
    },
    access                  [6]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      object-invalidated              (1),
                                      hardware-fault                  (2),
                                      object-access-denied            (3),
                                      invalid-address                 (4),
                                      object-attribute-inconsistent   (5),
                                      object-access-unsupported       (6),
                                      object-non-existent             (7),
                                      type-conflict                   (8),
                                      named-access-unsupported        (9),
                                      access-to-element-              (10)
                                      unsupported
    },
    objectDescription       [7]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      name-length-overflow            (1),
                                      od-overflow                     (2),
                                      od-write-protected              (3),
                                      extension-length-overflow       (4),
                                      od-description-length-          (5),
                                      overflow
                                      operational-problem             (6)
    },
    conclude                [9]   IMPLICIT  Integer8 {
                                      other                           (0),
                                      further-communication-          (1)
                                      required
    },
    other                   [8]   IMPLICIT  Integer8 {
                                      other                           (0)
    }
}
```

4.2.8 PDU non confrmées

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

UnconfirmedSendPD-PDU ::= BIT STRING
```

4.2.9 ASE Management

4.2.9.1 Service Get attributes

```
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 ASE Application process (processus d'application)

4.2.10.1 Service Get status

```
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 Service Identify

```
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 ASE Function invocation (Invocation de fonction)

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 Service Information report (rapport d'informations)

```
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 Service "Write"

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

Write-ResponsePDU ::= NULL
```

4.3 Définitions des types des ASE

4.3.1 Types de l'ASE AP

4.3.1.1 Ap_AccessProtectionSupported

```
Ap_AccessProtectionSupported ::= Boolean        -- True (vrai) signifie que la protection d'accès (Access
Protection) est prise en charge.
                                                -- False (faux) signifie que la protection d'accès (Access
Protection) n'est pas prise en charge.


Ap_AccessControl ::= BitString {                -- Le mot de passe Password (Unsigned8) est codé comme 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 de l'ASE AR

4.3.2.1 Reason Code

```
ReasonCode ::= Unsigned8
```

4.3.2.1.1 Additional Detail

```
AdditionalDetailI ::= OctetString
```

4.3.2.2 AREP

```
AddressAREP::= Unsigned8                -- Octet de poids faible de l'adresse de DLCEP
```

4.3.3 Types de l'ASE Function Invocation

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 codé comme 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 de l'ASE Management

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 de l'ASE Variable

4.3.5.1 Types généraux

4.3.5.1.1 Gn_Deletable

```
Gn_Deletable ::= Boolean          -- True (vrai) signifie "deletable" (effaçable).
                                  -- False (faux) signifie non effaçable.
```

4.3.5.1.2 Gn_Reusable

```
Gn_Reusable ::= Boolean           -- True (vrai)  signifie "reusable"  (réutilisable).
                                  -- False (faux) signifie non réutilisable.
```

4.3.5.1.3 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
}
```

4.3.5.1.4 Gn_Length

```
Gn_Length ::= Unsigned8
```

4.3.5.1.5 Gn_MoreFollows

```
Gn_MoreFollows ::= Boolean
```

4.3.5.1.6 Gn_Name

```
Gn_Name  ::=  OctetString
```

4.3.5.1.7 Gn_NumericID

```
Gn_NumericID  ::=  Unsigned16     -- Les valeurs de ce paramètre sont uniques au sein d'un AP.
```

4.3.5.1.8 Gn_ObjectDefinition

```
Gn_ObjectDefinition ::= OctetString     -- La sémantique de ce paramètre est spécifique à une application.
```

4.3.5.1.9 Gn_SubIndex

```
Gn_SubIndex ::= Unsigned8
```

4.3.5.2 Gn_ObjectClass

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

4.3.5.3 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
}
```

4.4 Définitions d'objet

4.4.1 Définition de haut niveau

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

4.4.2 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
}
```

4.4.3 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
   }
}
```

4.4.4 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
}
```

4.4.5 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
}
```

4.5 Syntaxe abstraite des types de données ("Data types")

4.5.1 Notation pour le type Boolean (booléen)

```
Boolean ::= BOOLEAN                 -- TRUE (vrai) si la valeur est non nulle.
                                    -- FALSE (faux) si la valeur est zéro.
```

4.5.2 Notation pour le type Integer (entier)

```
Integer ::= INTEGER                 -- n'importe quel entier
Integer8::= INTEGER  (-128..+127)   -- plage  -2^7 <= i <= 2^7-1
Integer16 ::= INTEGER  (-32768..+32767) -- plage  -2^15 <= i <= 215-1
Integer32 ::= INTEGER               -- plage  -2^31 <= i <= 2^31-1
```

4.5.3 Notation pour le type Unsigned (non signé)

```
Unsigned ::= INTEGER                -- tout entier non négatif
Unsigned8::= INTEGER  (0..255)      -- plage 0 <= i <= 2^8-1
Unsigned16::= INTEGER  (0..65535)   -- plage 0 <= i <= 2^16-1
Unsigned32::= INTEGER               -- plage 0 <= i <= 2^32-1
```

4.5.4 Notation pour le type Floating Point (virgule flottante)

```
Floating32 ::= BIT STRING  SIZE (4)     --  Simple précision IEC-60559
```

4.5.5 Notation pour le type BitString (chaîne de bits)

```
BitString ::= BIT STRING                -- Pour usage générique
```

4.5.6 Notation pour le type OctetString (chaîne d'octets)

```
OctetString ::= OCTET STRING            -- Pour usage générique
```

4.5.7 Notation pour le type VisibleString (chaîne visible)

```
VisibleString ::= VISIBLE STRING        -- Pour usage générique
```

4.5.8 Notation pour le type TimeOfDay (heure du jour)

```
TimeOfDay ::= OctetString6
```

4.5.9 Notation pour le type TimeDifference (différence horaire)

```
TimeDifference ::= OctetString6
```

4.5.10 Notation pour le type BinaryDate2000 (Date 2000 binaire)

BinaryDate2000 ::= OctetString8

5 Syntaxe de transfert

5.1 Peripherals encoding rules (règles de codage de périphériques (PER))

La Règle de codage de périphériques est le codage des types de PDU définis en 4.1.

5.2 Codage de types d'APDU

5.2.1 Vue d'ensemble du codage des APDU

Les APDU codées avec la PER doivent avoir un format uniforme. Les APDU doivent être constituées de deux parties principales, la partie “APDU Header” (en-tête d'APDU) et la partie “APDU Body” (corps d'APDU) conformément à la Figure 1.

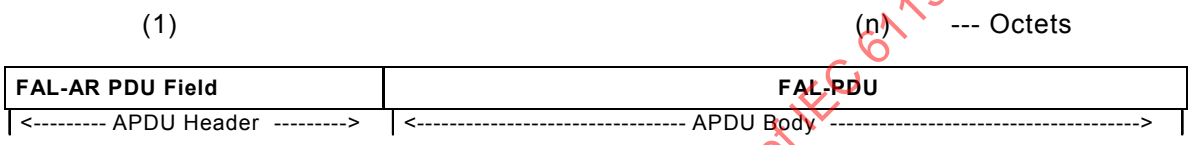


Figure 1 – Vue d'ensemble d'une APDU

5.2.2 Codage de l'en-tête d'APDU

En 4.1, un en-tête d'APDU est défini comme étant un type marqué de la classe d'étiquette 'APPLICATION'.

L'en-tête d'APDU (également appelé Champ APU AR de la FAL) doit être codé en utilisant un ou plusieurs octets tels que décrits dans le présent paragraphe.

Au sein de l'en-tête d'APDU, l'étiquette du type de PDU (par exemple: Establish-RequestPDU, ConfirmedSend-RequestPDU...) et les informations d'adresse (AddressAREP) sont codées.

Si la valeur du type de PDU est < 3, l'en-tête de l'APDU doit être codé en utilisant un seul octet tel que défini à la Figure 2.

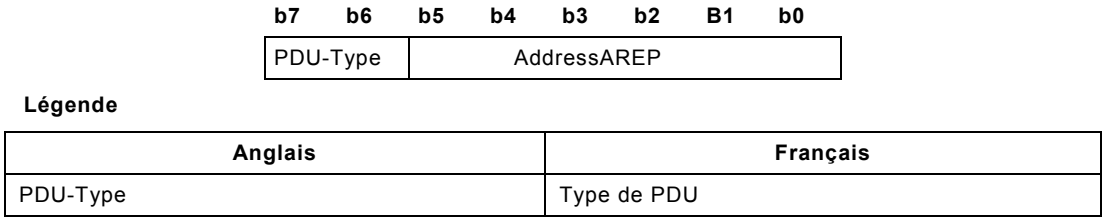
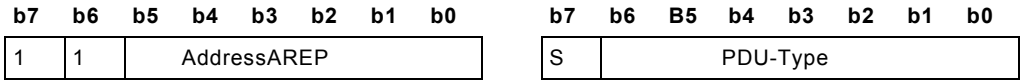


Figure 2 – En-tête d'APDU

Si la valeur du type d'APDU est >= 3, un octet d'extension est utilisé pour le codage de l'étiquette. L'utilisation d'un octet d'extension d'étiquette doit être indiquée en mettant à 1 les deux bits dans le champ Type de PDU du premier octet. Si les informations d'adresse (AddressAREP) sont supérieures à 62, un octet d'extension d'adresse doit être utilisé pour coder la valeur d'adresse. Dans ce cas, l'utilisation d'une extension d'adresse doit être indiquée en mettant à 1 tous les bits du champ d'adresse dans le premier octet (voir Figure 3 et Figure 4).

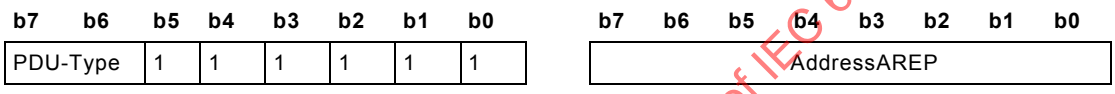


Légende

Anglais	Français
PDU-Type	Type de PDU

Figure 3 – PDU avec extension de type

Le champ S est réservé pour les extensions de gestion du système. Pour les PDU de la FAL, ce fanion est mis à 0.

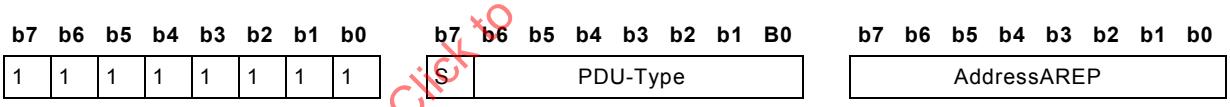


Légende

Anglais	Français
PDU-Type	Type de PDU

Figure 4 – PDU avec extension d'adresse

La Figure 5 montre le codage d'un en-tête d'APDU utilisant l'extension d'étiquette et l'octet d'extension d'adresse.



Légende

Anglais	Français
PDU-Type	Type de PDU

Figure 5 – PDU avec extension de type et de longueur

Codage spécial du type de PDU Establish-Request:

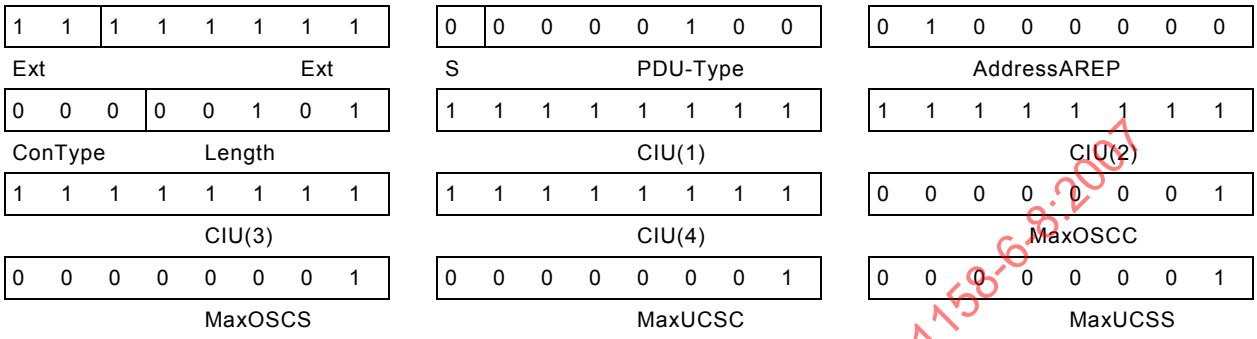
Le codage de la PDU Establish-Request dffère des règles décrites ci-dessus, car dans cette PDU, des paramètres complémentaires contenant le contexte d'AR sont passés. Les paramètres sont le type de connexion (connexion maître-maître pour transfert de données acyclique), le paramètre CIU (Control Interval User-triggered (intervalle de commande déclenché par un utilisateur)) et les compteurs de services en cours. Le paramètre type de connexion est codé dans les trois bits de poids fort. Les cinq bits de poids faible contiennent le nombre des paramètres d'établissement complémentaires.

Si l'étiquette pour l'adresse exige une extension (le type de PDU Establish a une extension), une longueur de PDU minimale de 9 octets est utilisée dans une demande de service Establish (voir Figure 6).

Exemple pour Establish-RequestPDU:

Étiquette d'Establish-RequestPDU : 4

AddressAREP : 64
ConnectionType : 0 pour MMAZ
CIU : FFFFFFFF
S : 0
MaxOSCC : 1
MaxOSCS : 1
MaxUCSC : 1
MaxUCSS : 1



Légende

Anglais	Français
PDU-Type	Type de PDU
Length	Longueur

Figure 6 – Exemple d'une PDU Establish-Request

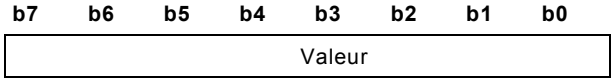
5.2.3 Codage de corps d'APDU

Le format du corps d'APDU est décrit en 4.1 en utilisant la syntaxe ASN.1. Dans les articles suivants, le codage des éléments de langage utilisés est décrit.

5.3 Codage des valeurs de type marqué

5.3.1 Codages des valeurs du type marqué de la classe d'étiquette PRIVATE

Les valeurs de type marqué (Tagged) avec des étiquettes de classe PRIVATE doivent être codées de la même manière que leurs types de base. Les étiquettes de classe PRIVATE ne sont pas codées explicitement (voir Figure 7). Elles sont utilisées uniquement pour atteindre la justesse syntaxique des définitions de l'ASN.1.



Légende

Anglais	Français
Value	Valeur

Figure 7 – Codage d'une valeur marquée PRIVATE

Les restrictions suivantes existent pour l'utilisation des types marqués PRIVATE.

– La longueur et la définition du codage d'un type marqué PRIVATE ne doivent pas varier.
– Dans une construction SEQUENCE, un seul type marqué PRIVATE au maximum peut être OPTIONAL.
– Un type marqué private ne doit pas être contenu dans une construction CHOICE.

5.3.2 Codage des valeurs de type marqué spécifiques à un contexte

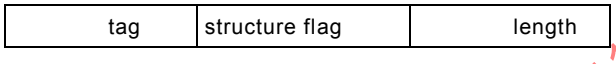
5.3.2.1 Vue d'ensemble

Les valeurs de type marqué spécifiques à un contexte sont codées comme une séquence d'un ou plusieurs octets d'informations d'identification (Identification information (ID-info)), suivie d'un ou plusieurs octets Value (ContentsOctets) conformément aux articles suivants. Le format résultant est montré à la Figure 8.

Figure 8 – Codage d'une valeur marquée spécifique à un contexte

5.3.2.2 Codage des informations d'identification (Identification information (ID-info))

L'ID-info est composé de trois champs: le champ "tag" (étiquette), le champ "structure flag" (fanion de structure) et le champ "length" (longueur) (voir Figure 9).



Légende

Anglais	Français
Tag	Étiquette
Structure flag	Fanion de structure
Length	Longueur

Figure 9 – Champs d'informations d'identification

Le champ "longueur" contient soit le nombre des octets de données subséquents, soit le nombre des valeurs suivantes, conformément au présent paragraphe.

Si le fanion de structure est mis à 1, il indique qu'une structure suit. Dans le codage de structure, chaque valeur est identifiée par son propre ID-info. Dans ce cas, le nombre des valeurs dans la structure est codé dans le champ length de l'ID-info. Dans le champ length de la dernière valeur de la structure, la longueur du codage en octets des valeurs est contenue.

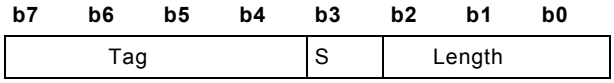
Si le fanion de la structure est 0, et la valeur ne fait pas partie d'une structure comme décrit avant, les valeurs qui suivent ne sont pas marquées. Dans ce cas, les types et les longueurs des valeurs contenues sont connues implicitement; le nombre des valeurs contenues (paramètres) est codé dans le champ length.

Si le fanion de la structure est 0, et la valeur ne fait pas partie d'une autre structure, le champ length doit contenir:

le nombre des valeurs qui suivent, si cette valeur n'est pas la dernière valeur dans la structure,

le nombre d'octets de codage de la valeur, si cette valeur est la dernière valeur dans la structure.

Le champ "tag" (étiquette) contient la valeur de l'étiquette définie dans la définition de syntaxe de l'ASN.1. Si la valeur de l'étiquette est < 15 et la valeur "length" est < 7, l'ID-info doit être codé en utilisant un(1) octet (voir Figure 10).

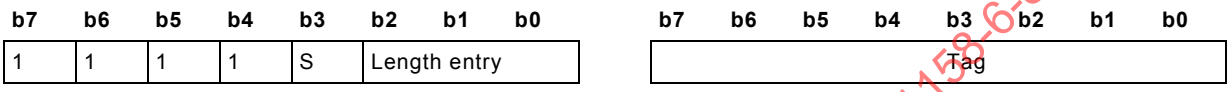


Légende

Anglais	Français
Tag	Étiquette
Length	Longueur

Figure 10 – ID-info pour tag 0 .. 14, length entry 0 .. 6

Si la valeur de l'étiquette est >= 15 et la valeur "length" est < 7, l'ID-info doit être codé en utilisant deux (2) octets (voir Figure 11).

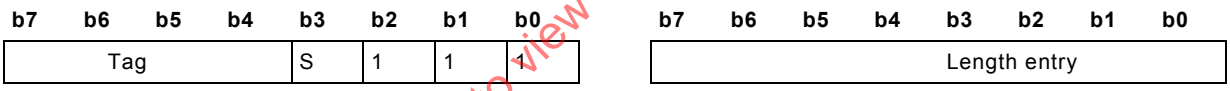


Légende

Anglais	Français
Length entry	Entrée de longueur
Tag	Étiquette

Figure 11 – ID-info pour tag 15 .. 255, length entry 0 .. 6

Si la valeur de l'étiquette est < 15 et la valeur "length" est >= 7, l'ID-info doit être codé en utilisant deux (2) octets (voir Figure 12).

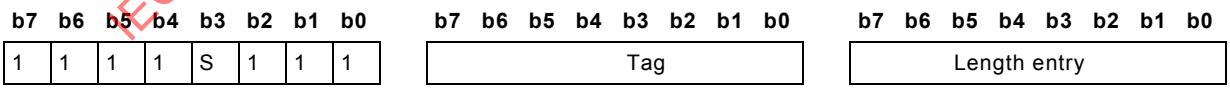


Légende

Anglais	Français
Tag	Étiquette
Length entry	Entrée de longueur

Figure 12 – ID-info pour tag 0 .. 14 , length entry 7 .. 255

Si la valeur de l'étiquette est >= 15 et la valeur "length" est >= 7, l'ID-info doit être codé en utilisant trois (3) octets (voir Figure 13).



Légende

Anglais	Français
Tag	Étiquette
Length entry	Entrée de longueur

Figure 13 – ID-info pour tag 15 .. 255, length entry 7 .. 255

5.3.2.3 Exemple

"Data" de type de données "Boolean", par exemple, est codé comme montré à la Figure 14 et à la Figure 15:

b7	b6	b5	b4	b3	b2	b1	b0
Tag				0	0	0	1

b7	b6	b5	b4	b3	b2	b1	b0
1	1	1	1	1	1	1	1

Légende

Anglais	Français
Tag	Étiquette

Figure 14 – Codage de la valeur booléenne TRUE

b7	b6	b5	b4	b3	b2	b1	b0
Tag				0	0	0	1

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0

Légende

Anglais	Français
Tag	Étiquette

Figure 15 – Codage de la valeur booléenne FALSE

5.4 Codage des valeurs simples

5.4.1 Codage d'une valeur Boolean

1) Le codage d'une valeur de type Boolean doit être primitif, et les ContentsOctets doivent consister exactement en un octet.

2) La valeur TRUE est codée en mettant tous les bits de l'octet à 1, la valeur FALSE est codée en mettant tous les bits de l'octet à 0.

5.4.2 Codage de valeurs String (chaîne visible, chaîne d'octets, chaîne de bits)

Dans le codage de chaînes, les entrées de longueur (length entries), qui donnent le nombre d'octets suivants, sont placées dans le premier octet des éléments individuels de chaîne (voir Figure 16). Les données d'utilisateur sans une étiquette sont fournies sans ID-info. Les données sont identifiées par leur position dans la PDU et la longueur est implicitement connue et fixée.

Octet 1	Octet 2	Octet 3		Octet n + 1
Length = n	Element 1	Element 2		Element n

Légende

Anglais	Français
Length = n	Longueur = n
Element 1	Élément 1
Element 2	Élément 2
Element n	Élément n

Figure 16 – Codage de Strings (chaînes)

5.4.3 Codage d'une valeur Integer

a) Le codage d'une valeur Integer de longueur fixe des types Integer8, Integer16 et Integer32 doit être primitif et les ContentsOctets doivent respectivement consister exactement en un octet, deux octets et quatre octets.

b) Les ContentsOctets doivent être un nombre binaire en complément à deux égal à la valeur entière et consister en les bits 8 à 1 du premier octet, suivis des bits 8 à 1 du deuxième octet, suivis des bits 8 à 1 de chaque octet à tour de rôle jusqu'au dernier octet inclus des ContentsOctets.

5.4.4 Codage d'une valeur Unsigned

a) Le codage d'une valeur Unsigned de longueur fixe des types Unsigned8, Unsigned16 et Unsigned32 doit être primitif et les ContentsOctets doivent consister exactement en un octet, deux octets et quatre octets respectivement.

b) Les ContentsOctets doivent être un nombre binaire égal à la valeur Unsigned et consister en les bits 8 à 1 du premier octet, suivis des bits 8 à 1 du deuxième octet, suivis des bits 8 à 1 de chaque octet à tour de rôle jusqu'au dernier octet inclus des ContentsOctets.

5.4.5 Codage d'une valeur Floating-point

a) Le codage d'une valeur Floating32 doit être primitif et les ContentsOctets doivent consister exactement en quatre ou huit octets, respectivement.

b) Les ContentsOctets doivent contenir des valeurs en virgule flottante définies conformément à la Norme ANSI/IEEE 754. Le signe est codé dans le bit 8 du premier octet. Il est suivi de l'exposant commençant à partir du bit 7 du premier octet et ensuite de la mantisse commençant à partir du bit 7 du deuxième octet pour le type Floating32.

5.4.6 Codage d'une valeur BinaryDate

a) Le codage d'une valeur BinaryDate doit être primitif.

b) Le champ Length doit indiquer comme un nombre binaire, le nombre d'octets dans la valeur BinaryDate.

c) Les ContentsOctets doivent avoir une valeur égale à celle des octets dans la valeur de données, comme montré à la

Anglais	Français
0…59 min	0…59 minutes
0…23 hours	0…23 heures
day of week	Jour de la semaine
day of month	Jour du mois
1…7 d. of w.	1…7 jours de la semaine
1…31 d. of m.	1…31 jours du mois.
1…12 months	1…12 mois
0 … 99 years	0 … 99 ans
msb	Bit de poids fort (msb)

Figure 17.

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 ms …59 999 ms
3	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 min
4	SU	0	0	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…23 hours
	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	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…12 months
7	0	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0 … 99 years
	msb								

Anglais	Français
0…59 min	0…59 minutes
0…23 hours	0…23 heures
day of week	Jour de la semaine
day of month	Jour du mois
1…7 d. of w.	1…7 jours de la semaine
1…31 d. of m.	1…31 jours du mois.
1…12 months	1…12 mois
0 … 99 years	0 … 99 ans
msb	Bit de poids fort (msb)

Figure 17 – Codage d'une valeur BinaryDate

5.4.7 Codage d'une valeur BinaryDate2000

a) Le codage d'une valeur BinaryDate2000 doit être primitif.

b) Le champ Length doit indiquer en tant qu'un nombre binaire, le nombre d'octets dans la valeur BinaryDate2000.

c) Les ContentsOctets doivent avoir une valeur égale à celle des octets dans la valeur de données, comme montré à la Figure 18.

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 ms …59 999 ms
3	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…59 min
4	SU	0	0	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0…23 hours
	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	0	0	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1…12 months
7	0	0	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	0 … 9999 years
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	with century
	msb								

Légende

Anglais	Français
0…59 min	0…59 minutes
0…23 hours	0…23 heures
day of week	Jour de la semaine
day of month	Jour du mois
1…7 d. of w.	1…7 jours de la semaine
1…31 d. of m.	1…31 jours du mois.
1…12 months	1…12 mois
0 … 9999 years	0 … 9999 ans
with century	Avec siècle

Anglais	Français
msb	Bit de poids fort (msb)

Figure 18 – Codage d'une valeur BinaryDate2000

5.4.8 Codage d'une valeur Time-of-day

a) Le codage d'une valeur Time Of Day doit être primitif.

b) Le champ Length doit indiquer en tant qu'un nombre binaire, le nombre d'octets dans la valeur Time Of Day.

c) Les ContentsOctets doivent avoir une valeur égale à celle des octets dans la valeur de données, comme montré à la Figure 19.

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 01.01.84
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	optional
	msb								

Légende

Anglais	Français
number of milliseconds since midnight	Nombre de millisecondes depuis minuit
number of days since 01.01.84	Nombre de jours depuis 01.01.84
optional	Facultatif
msb	Bit de poids fort (msb)

Figure 19 – Codage d'une valeur Time-of-day

5.4.9 Codage d'une valeur Time-difference

a) Le codage d'une valeur Time Difference doit être primitif.

b) Le champ Length doit indiquer comme un nombre binaire, le nombre d'octets dans la valeur Time Difference.

c) Les ContentsOctets doivent avoir une valeur égale à celle des octets dans la valeur de données, comme montré à la Figure 20.

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								

Légende

Anglais	Français
number of milliseconds	Nombre de millisecondes
number of days	Nombre de jours
optional	Facultatif
msb	Bit de poids fort (msb)

Figure 20 – Codage d'une valeur Time-difference

5.4.10 Codage d'une valeur Time

a) Le codage d'une valeur Time doit être primitif.

b) Le champ Length doit indiquer comme un nombre binaire, le nombre d'octets dans la valeur Time.

c) Les ContentsOctets doivent avoir une valeur égale à celle des octets dans la valeur de données, comme montré à la Figure 21.

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}	signed integer
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	of 8 octet length of 1/32ms unit
	MSB								

Légende

Anglais	Français
signed integer	Entier de type «Signed»
of 8 octet length of 1/32ms unit	De longueur 8 octets d'unité 1/32ms
MSB	Bit de poids fort (MSB)

Figure 21 – Codage d'une valeur Time

5.4.11 Codage d'une valeur Null

a) Le codage d'une valeur NULL doit être primitif.

b) Le ContentsOctet doit être omis.

5.4.12 Codage d'une valeur ANY

Le type de données ANY contient un ou plusieurs éléments de données des types de données qui sont mis en chaîne ensemble sans aucun espace. La composition est connue implicitement.

5.4.13 Codage de types structurés

Lorsqu'un type structuré est codé, le fanion de structure de l'Identification Information doit être mis à un. Le champ Length, qu'il soit ou ne soit pas étendu, doit contenir le nombre de composants codés du type structuré.

5.4.14 Codage d'une valeur SEQUENCE

Le type SEQUENCE est comparable à un enregistrement. Il représente une collection de données d'utilisateurs de mêmes ou de différents types de données.

Une valeur de type SEQUENCE peut contenir une variable simple ou une variable structurée supplémentaire comme ses composants. Si un type SEQUENCE contient une autre valeur de type structuré, il doit être compté comme un composant unique même s'il contient plusieurs composants.

5.4.15 Codage d'une valeur SEQUENCE OF

Le type SEQUENCE OF représente une succession de composants. Il est comparable à une matrice (array).

Une valeur de type SEQUENCE OF peut contenir une ou plusieurs variables simples ou construites. Si un type SEQUENCE OF contient une autre valeur de type structuré, il doit être compté comme un composant unique même s'il contient plusieurs composants.

Le codage est comme pour la structure SEQUENCE. Pour l'indication du nombre de composants, le nombre de répétitions doit être pris en compte.

5.4.16 Codage d'une valeur CHOICE

Un type CHOICE représente une sélection d'un ensemble de valeurs prédéfinies. Les composants d'une construction CHOICE doivent avoir différentes étiquettes afin de permettre une identification correcte. Au lieu de la construction CHOICE, le composant réellement sélectionné est codé. Un seul composant doit être codé pour une construction CHOICE.

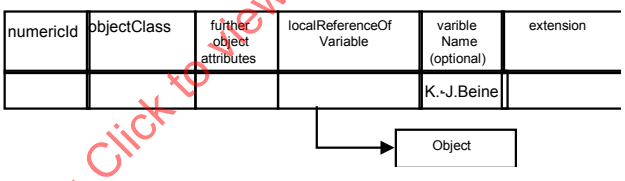
5.4.17 Codage d'une valeur ENUMERATED

Une valeur ENUMERATED doit être codée en tant qu'une valeur Unsigned8.

5.4.18 Codage d'une valeur Object-definition

Le codage des définitions d'objets individuels en paramètres "List Of Objects and Attributes" (type de données Gn_ObjectDefinition) des GetAttributePDUs est montré dans le présent paragraphe.

L'attribut objectClass est l'identificateur de l'objet et il indique la classe à laquelle appartient cet objet. Les autres attributs d'objets sont spécifiques aux objets et doivent être codés en tant qu'une chaîne d'octets. En plus des attributs d'objets, la classe d'objets doit être émise dans le service GetAttributes, sauf pour la List Header (en-tête de liste), dont le numeric ID est zéro. Un exemple de la structure d'un Object Definition est illustré dans la Figure 22.



Légende

Anglais	Français
Further objet attributes	Plus d'attributs d'objets
Variable Name (optional)	Nom de variable (facultatif)
Extension	Extension
Object	Objet

Figure 22 – Exemple pour un Object definition

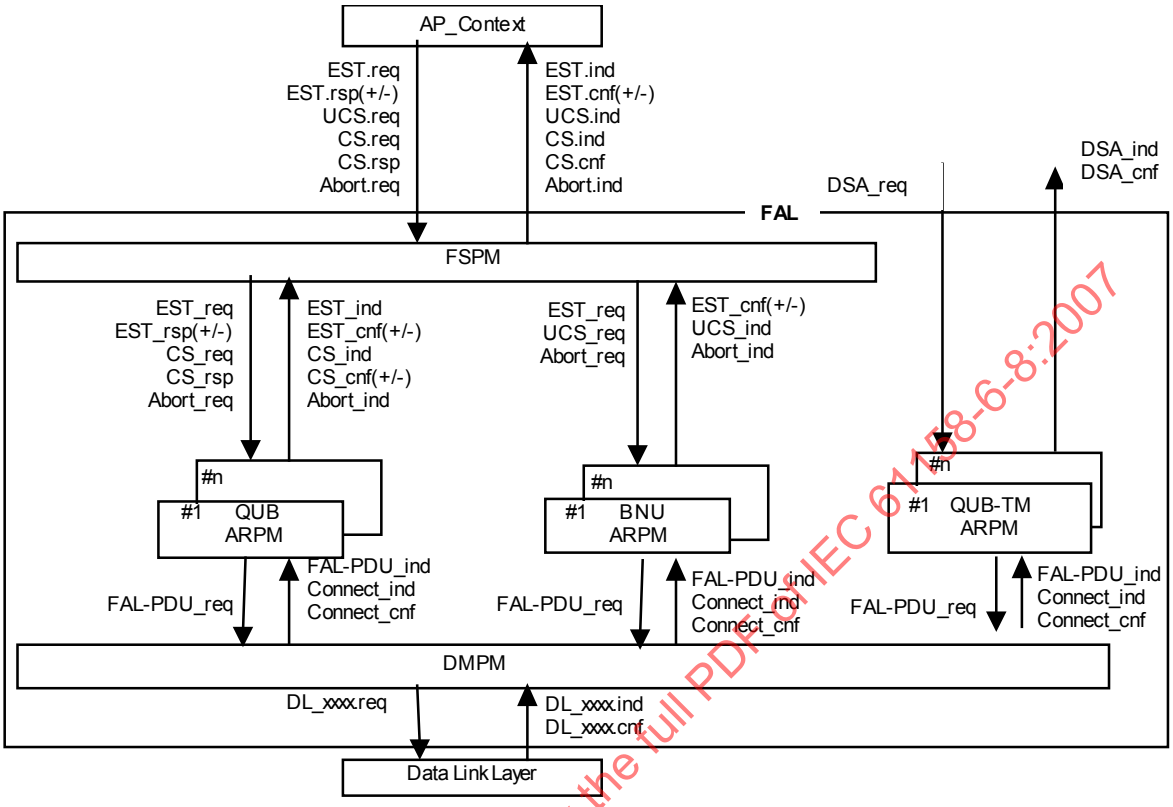
6 Vue d'ensemble d'un diagramme protocolaire

Les diagrammes de protocole et de services d'interfaces à la FAL sont spécifiés dans le présent paragraphe.

NOTE Les diagrammes d'états spécifiés dans le présent paragraphe et les ARPM définis dans les articles suivants définissent seulement les événements valides pour chacun. Il s'agit d'une affaire locale pour gérer les événements non valides.

Le comportement de la FAL est décrit par trois diagrammes protocolaires intégrés. Des ensembles spécifiques de ces diagrammes protocolaires sont définis pour différents types d'AREP. Les trois diagrammes protocolaires sont: FAL Service Protocol Machine (Machine de protocole de service FAL (FSPM)), Application Relationship Protocol Machine (Machine de

protocole de relations AR (ARPM)), et Data Link Layer Mapping Protocol Machine (Machine de protocole de mapping de couche Liaison de Données (DMPM)). La Figure 23 montre les primitives échangées entre les diagrammes protocolaires.



Légende

Anglais	Français
Data Link Layer	Couche liaison de données

Figure 23 – Primitives échangées entre les diagrammes protocolaires

7 Diagramme d'états AP-Context (contexte d'AP)

7.1 Définitions des primitives

7.1.1 Primitives échangées entre l'utilisateur de la FAL et l'AP-Context

Les primitives échangées entre l'utilisateur de la FAL et l'AP-Context sont montrées dans le Tableau 1 et le Tableau 2.

Tableau 1 – Primitives émises par l'utilisateur de la FAL vers l'AP-context

Nom de primitive	Source	Paramètres et fonctions associés
Terminate.req	Utilisateur de la FAL	Se référer à la définition du service de la FAL (61158-508)
Initiate.req	Utilisateur de la FAL	
Initiate.rsp(+)	Utilisateur de la FAL	
Initiate.rsp(-)	Utilisateur de la FAL	
ConfirmedService.req	Utilisateur de la FAL	
ConfirmedService.rsp	Utilisateur de la FAL	
UnconfirmedService.req	Utilisateur de la FAL	

Tableau 2 – Primitives émises par l'AP-context vers l'utilisateur de la FAL

Nom de primitive	Source	Paramètres et fonctions associés
Terminate.ind	AP-Context	Se référer à la définition du service de la FAL (61158-508)
Initiate.ind	AP-Context	
Initiate.cnf(+)	AP-Context	
Initiate.cnf(-)	AP-Context	
ConfirmedService.ind	AP-Context	
ConfirmedService.cnf	AP-Context	
UnconfirmedService.ind	AP-Context	

7.2 Description de diagramme d'états

Les attributs utilisés dans ce diagramme d'états sont définis comme des éléments de l'attribut de l'AP "*List Of In-Use AR Endpoint Info"*. Voir Figure 24 pour le diagramme d'états.

CLOSED (fermé)

L'AP-Context pour une AR n'est pas ouvert. Seules les primitives de service Initiate.req, Establish.ind, Terminate.req et Abort.ind sont autorisées. Toutes les autres primitives de service doivent être rejetées avec le service Terminate.

OPENING-REQUESTING (REQ)

L'utilisateur de la FAL locale souhaite ouvrir l'AP-Context pour une AR. Seules les primitives de service Establish.cnf(+), Establish.cnf(-), Terminate.req et Abort.ind sont autorisées. Tous les autres services doivent être rejetés avec le service Terminate.

OPENING-RESPONDING (RSP)

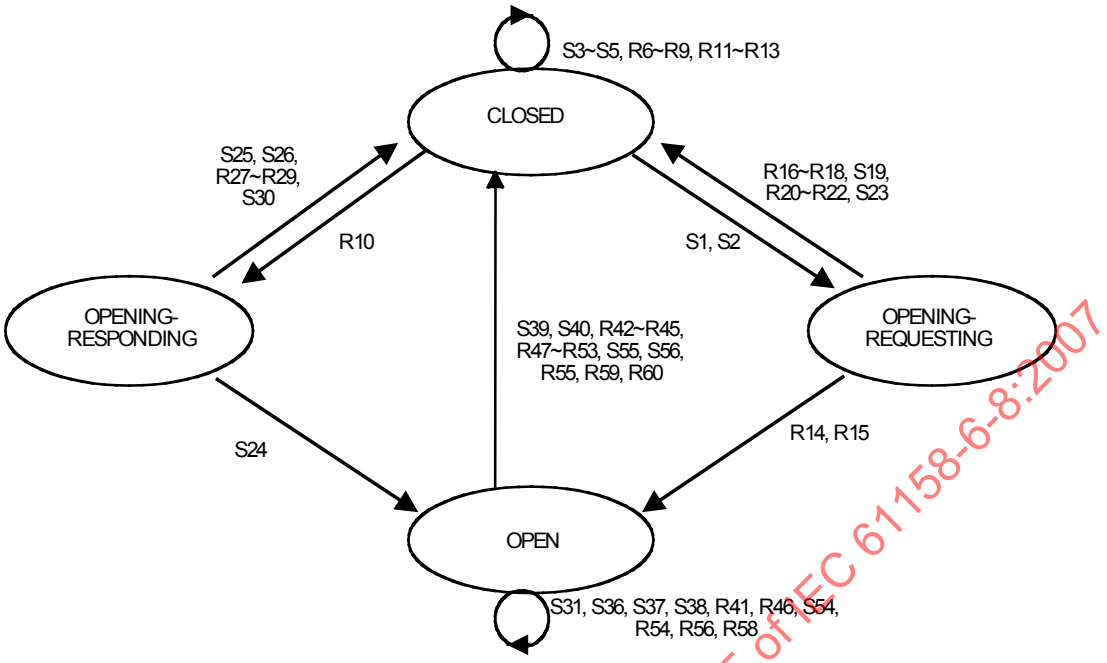
L'AP-Context distant souhaite ouvrir l'AP-Context pour une AR. Seules les primitives de service Initiate.rsp(+), Initiate.rsp(-), Terminate.req et Abort.ind sont autorisées. Toutes les autres primitives de service doivent être rejetées avec le service Terminate.

OPEN (ouvert)

L'AP-Context pour une AR est ouvert. Les primitives de service Initiate.req, Initiate.rsp(+), Initiate.rsp(-) ne sont pas autorisées et doivent être rejetées avec le service Terminate. Les actions suivantes doivent être prises pour réinitialiser l'AP-Context pour une AR.

Mettre à zéro le Compteur "*Outstanding Services Requesting*" et le Compteur "*Outstanding Services Responding"*.

Mettre l'état du Contexte à "CLOSED"



Légende

Anglais	Français
OPEN	OUVERT
CLOSED	FERMÉ
OPENING-RESPONDING	OUVRANT-RÉPONDANT
OPENING-REQUESTING	OUVRANT-DEMANDANT

Figure 24 – Diagramme d'états d'initiation d'un AP vers un AP-context

7.3 Transitions d'états d'initiation d'un AP vers un AP-context

Les transitions d'états d'initiation d'un AP vers un AP-context sont définies dans le Tableau 3 et le Tableau 4. Dans les tableaux du présent paragraphe, le paramètre "Data" pour la Machine de Protocole de Service FAL s'exprime par "FalApduBody" afin de mettre en relief sa sémantique.

Tableau 3 – Transactions de l'émetteur du diagramme d'états de l'AP-context

#	État courant	Événement ou condition => action	Prochain État

S1	CLOSED	Initiate.req
		&& ApContextTest = "True"
		&& ApExplicitConnection = "True"
		=>
		EST.req {
		FalApduBody := "Establish-CommandPDU"
		}
		or
		EST.req {
		FalApduBody := "Establish-RequestPDU"
		}
		or
		EST.req {
		FalApduBody := "Establish-Request2PDU"
		}
			REQ

S2	CLOSED	Initiate.req
		&& ApContextTest = "True"
		&& ApExplicitConnection = "False"
		=>
		EST.req {
		FalApduBody := "NULL"
		}
			REQ

S3	CLOSED	Initiate.req
		&& ApContextTest = ""False""
		=>
		Terminate.ind {
		LocallyGenerated := "True",
		TerminateIdentifier := "FAL",
		ReasonCode := "Context error"
		}
			CLOSED

S4	CLOSED	Terminate.req => (pas d'actions prises)	CLOSED

S5	CLOSED	FAL service.primitive <> "Initiate"
		&& FAL service.primitive <> "Terminate"
		=>
		Terminate.ind {
		LocallyGenerated := "True",
		TerminateIdentifier := "FAL",
		ReasonCode := "User error"
		}
			CLOSED

S19	REQ	Terminate.req => Abort.req { Originator:= "TerminateIdentifier of Terminate.req", ReasonCode := "ReasonCode of Terminate.req" }, ResetArep	CLOSED

S23	REQ	FAL service.primitive <> "Initiate.rsp""
		&& FAL service.primitive <> "Terminate.req"
		=>
		Abort.req {
		Originator := "FAL",
		ReasonCode := "User error"
		},
		
		Terminate.ind {
		LocallyGenerated := "True",
		TerminateIdentifier := "FAL",
		ReasonCode := "User error"
		},
		
		ResetArep
			CLOSED

#	État courant	Événement ou condition => action	Prochain État

S24	Rsp	Initiate.rsp(+)
		=>
		EST.rsp(+) {
		FalApduBody := “Establish-AffirmativePDU”
		}
		or
		EST.rsp(+) {
		FalApduBody = “Establsih-ResponsePDU”
		}
			OPEN

S25	Rsp	Initiate.rsp(-)
		=>
		EST.rsp(-){
		FalApduBody := “Establish-NegativePDU”
		},
		or
		EST.rsp(-) {
		FalApduBody = “Establsih-ErrorPDU”
		},
		
		ResetArep
			CLOSED

S26	Rsp	Terminate.req => Abort.req { Originator := “TerminateIdentifier of Terminate.req”, ReasonCode := “ReasonCode of Terminate.req” }, ResetArep	CLOSED

S30	Rsp	FAL service.primitive <> “Initiate.rsp”
		&& FAL service.primitive <> “Terminate.req”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “User error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “User error”
		},
		
		ResetArep
			CLOSED

S31	OPEN	ConfirmedService.req && ConfirmedServiceCheck = “True” && OutstandingServicesRequestingCounter < NegotiatedMaxOutstanding-ServicesRequesting && InvokeIdExistent = “False” && PDU length ≤ NegotiatedMaxPduSizeSending && RequestedServiceSupportedTest = “True” => CS.req { FalApduBody := “ConfirmedSend-CommandPDU” }, or CS.req { FalApduBody := “ConfirmedSend-RequestPDU” }, OutstandingServicesRequestingCounter := OutstandingServicesRequestingCounter+1	OPEN

S36	OPEN	UnconfirmedService.req
		&& UnconfirmedServiceCheck = “True”
		&& PDU length ≤ NegotiatedMaxPduSizeSending
		&& RequestedServiceSupportedTest = “True”
		&& ImmediateAcknowledge = “False”
		=>
		UCS.req {
		FalApduBody := “UnconfirmedSend-CommandPDU”
		}
		or
		UCS.req {
		FalApduBody := “UnconfirmedSend-PDU”
		}
			OPEN

#	État courant	Événement ou condition => action	Prochain État

S37	OPEN	UnconfirmedService.req
		&& UnconfirmedServiceCheck = “True”
		&& PDU length ≤ NegotiatedMaxPduSizeSending
		&& RequestedServiceSupportedTest = “True”
		&& ImmediateAcknowledge = “True”
		=>
		UCA.req {
		FalApduBody := “UnconfirmedAcknowledgedSend-CommandPDU”
		}
		or
		UCA.req {
		FalApduBody := “UnconfirmedAcknowledgedSend-RequestPDU”
		}
			OPEN

S38	OPEN	AR_ASE service request && ArServiceCheck = “True” && RequestedServiceSupportedTest = “True” => ArFspmService NOTE Cet état est fourni pour accéder au FSPM directement à partir de l'utilisateur de la FAL. La fonction ArFspmService génère une primitive FSPM telle que définie ultérieurement dans le présent paragraphe.	OPEN
S39	OPEN	Terminate.req => Abort.req { Originator := “TerminateIdentifier of Terminate.req”, ReasonCode := “ReasonCode of Terminate.req” }, ResetArep	CLOSED

S40	OPEN	Service.primitive de la FAL erronée, inconnue ou non autorisée
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “User error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “User error”
		},
		
		ResetArep
			CLOSED

S54	OPEN	ConfirmedService.rsp && ConfirmedServiceCheck = “True” && InvokeIdExistent = “True” && SameService = “True” && PDU length ≤ NegotiatedMaxPduSizeSending => CS.rsp { FalApduBody := “ConfirmedSend-AffirmativePDU” }, or CS.rsp { FalApduBody := “ConfirmedSend-NegativePDU” }, or CS.rsp { FalApduBody := “ConfirmedSend-ResponsePDU” }. OutstandingServicesRespondingCounter := OutstandingServicesRespondingCounter-1 NOTE Un moyen de protocole supplémentaire doit être fourni pour déterminer si ConfirmedSend-AffirmativePDU ou ConfirmedSend-NegativePDU est utilisée.	OPEN

#	État courant	Événement ou condition => action	Prochain État

S55	OPEN	ConfirmedService.rsp
		&& ConfirmedServiceCheck = “True”
		&& InvokeIdExistent = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “InvokeID-error-response”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-response”
		},
		
		ResetArep
			CLOSED

S56	OPEN	ConfirmedService.rsp
		&& ConfirmedServiceCheck = “True”
		&& InvokeIdExistent = “True”
		&& SameService = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Service-error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Service-error”
		},
		
		ResetArep
			CLOSED

Tableau 4 – Transactions du récepteur du diagramme d’états de l'AP-context

#	État courant	Événement ou condition => action	Prochain État
R6	CLOSED	Abort.ind => (pas d'actions prises)	CLOSED

R7	CLOSED	Primitive de service AR_FSPM erronée ou inconnue
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR_ASE error”
		}
			CLOSED

R8	CLOSED	AR_FSPM service primitive <> “EST.ind”
		&& AR_FSPM service.primitive <> “Abort.ind”
		=>
		Abort.req {
		Originator:= “FAL”,
		ReasonCode := “Connection State Conflict”
		}
			CLOSED

R9	CLOSED	EST.ind
		&& FalApduBody = not allowed, unknown or faulty FAL PDU
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “FAL PDU error”
		}
			CLOSED

#	État courant	Événement ou condition => action	Prochain État

R10	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “True”
		&& MaxFalPduLengthTest = "True"
		&& ServicesSupportedTest = “True”
		=>
		NegotiateOutstandingServices,
		
		Initiate.ind {
		}
			Rsp

R11	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR error”
		}
			CLOSED

R12	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “True”
		&& MaxFalPduLengthTest = “False”
		=>
		EST.rsp(-) {
		FalApduBody := “Establish-NegativePDU”,
		ErrorCode := “Max-Fal-Fdu-Size-Insufficient”
		}
		or
		EST.rsp(-) {
		FalApduBody := “Establish-ErrorPDU”,
		ErrorCode := “Max-Fal-Fdu-Size-Insufficient”
		}
			CLOSED

R13	CLOSED	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		&& ApContextTest = “True”
		&& MaxFalPduLengthTest = “True”
		&& ServicesSupportedTest = “False”
		=>
		EST.rsp(-) {
		FalApduBody := “Establish-NegativePDU”,
		ErrorCode := “Service-Not-Supported”
		
		or
		
		EST.rsp(-) {
		FalApduBody := “Establish-ErrorPDU”,
		ErrorCode := “Service-Not-Supported”
		}
			CLOSED

R14	REQ	EST.cnf(+)
		&& (FalApduBody = “Establish-AffirmativePDU”
		|| FalApduBody = “Establish-ResponsePDU”)
		&& ApExplicitConnection = “True”
		=>
		Initiate.cnf(+) {
		}
			OPEN

R15	REQ	EST.cnf(+)
		&& FalApduBody = “NULL”
		&& ApExplicitConnection = “False”
		=>
		Initiate.cnf(+) {
		}
			OPEN

#	État courant	Événement ou condition => action	Prochain État
R16	REQ	EST.cnf(-) && (FalApduBody = “Establish-NegativePDU” || FalApduBody = “Establish-ErrorPDU”) && ApExplicitConnection = “True” => “Initiate.cnf(-) { ErrorCode := “Errorcode of EST.cnf(-)” }, ResetArep	CLOSED
R17	REQ	EST.cnf(-) && FalApduBody = “NULL” && ApExplicitConnection = “False” => Initiate.cnf(-) { ErrorCode := “Errorcode in EST.cnf” }, ResetArep	CLOSED
R18	REQ	Abort.ind => Terminate.ind { LocallyGenerated := “LocallyGenerated of Abort.ind”, TerminateIdentifier := “Originator of Abort.ind”, ReasonCode := “ReasonCode of Abort.ind” }, ResetArep	CLOSED

R20	REQ	Primitive de service AR_FSPM erronée ou inconnue
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		=>
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		ResetArep
			CLOSED

R21	REQ	AR_FSPM service.primitive <> “EST.cnf”
		&& AR_FSPM service.primitive <> “Abort.ind”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Connection State Conflict”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Connection State Conflict”
		},
		
		ResetArep
			CLOSED

R22	REQ	EST.cnf
		&& FalApduBody =“ not allowed, unknown or faulty FAL PDU”
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “FAL PDU error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “FAL PDU error”
		},
		
		ResetArep
			CLOSED

#	État courant	Événement ou condition => action	Prochain État
R27	Rsp	Abort.ind => Terminate.ind { LocallyGenerated := “LocallyGenerated of Abort.ind”, TerminateIdentifier := “Originator of Abort.ind”, ReasonCode := “ReasonCode of Abort.ind” }, ResetArep	CLOSED

R28	Rsp	Primitive de service AR_FSPM erronée ou inconnue
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		ResetArep
			CLOSED

R29	Rsp	AR_FSPM service primitive <> “ Abort.ind”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “State Conflict with AR_ASE”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “AP_ASE”,
		ReasonCode := “Connection State Conflict with AR_ASE”
		},
		
		ResetArep
			CLOSED

R41	OPEN	CS.ind && (FalApduBody = “ConfirmedSend-CommandPDU” || FalApduBody = “ConfimredSend-RequestPDU”) && PDU length ≤ NegotiatedMaxPduSizeReceiving && OutstandingServicesRespondingCounter < NegotiatedMaxOutstanding-ServicesResponding && InvokeIdExistent = “False” && IndicatedServiceSupportedTest = “True” => ConfirmedService.ind { }, OutstandingServicesRespondingCounter := OutstandingServicesRespondingCounter+1	OPEN

R42	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

#	État courant	Événement ou condition => action	Prochain État

R43	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& OutstandingServicesResponding ≥ NegotiatedMaxOutstanding-ServicesResponding
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Max-services-overflow”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Max-Services-Overflow”
		},
		
		ResetArep
			CLOSED

R44	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& OutstandingServicesRespondingCounter < NegotiatedMaxOutstanding-ServicesResponding
		&& InvokeIdExistent = “True”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “InvokeID-error-request”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-request”
		},
		
		ResetArep
			CLOSED

R45	OPEN	CS.ind
		&& (FalApduBody = “ConfirmedSend-CommandPDU”
		|| FalApduBody = “ConfimredSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& OutstandingServicesRespondingCounter < NegotiatedMaxOutstandingServicesResponding
		&& InvokeIdExistent = “False”
		&& IndicatedServiceSupportedTest = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		ResetArep
			CLOSED

R46	OPEN	UCS.ind
		&& (FalApduBody = “UnconfirmedSend-CommandPDU”
		|| FalApduBody = “UnconfimredSend-PDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& IndicatedServiceSupportedTest = “True”
		=>
		UnconfirmedService.ind {
		}
			OPEN

#	État courant	Événement ou condition => action	Prochain État

R47	OPEN	UCS.ind
		&& (FalApduBody = “UnconfirmedSend-CommandPDU”
		|| FalApduBody = “UnconfimredSend-PDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

R48	OPEN	UCS.ind
		&& (FalApduBody = “UnconfirmedSend-CommandPDU”
		|| FalApduBody = “UnconfimredSend-PDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& IndicatedServiceSupportedTest = “False”
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Feature-not-supported”
		},
		
		ResetArep
			CLOSED

R49	OPEN	Abort.ind => Terminate.ind { LocallyGenerated := “LocallyGenerated of Abort.ind”, TerminateIdentifier := “Originator of Abort.ind”, ReasonCode := “ReasonCode of Abort.ind” }, ResetArep	CLOSED

R50	OPEN	EST.ind
		&& (FalApduBody = “Establish-CommandPDU”
		|| FalApduBody = “Establish-RequestPDU”
		|| FalApduBody = “Establish-Request2PDU”)
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		ResetArep
			CLOSED

#	État courant	Événement ou condition => action	Prochain État

R51	OPEN	Primitive de service AR_FSPM erronée ou inconnue
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “AR_ASE error”
		},
		
		ResetArep
			CLOSED

R52	OPEN	Primitive de service AR_FSPM non autorisée
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		ResetArep
			CLOSED

R53	OPEN	Primitive de service Send AR_FSPM valide (l'une de CS, US, UCA)
		&& FalApduBody = not-allowed, unknown or faulty FAL PDU
		&& ArAccessSupported = “False”
		=>
		Abort.req {
		Originator := “AP_ASE”,
		ReasonCode := “FAL-PDU-error”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “FAL –PDU-error”
		},
		
		ResetArep
			CLOSED

R54	OPEN	UCA.ind
		&& ArAccessSupported = “True”
		&& (FalApduBody = “UnconfirmedAcknowledgedSend-CommandPDU”
		|| FalApduBody = “UnconfirmedAcknowledgedSend-RequestPDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		=>
		UnconfirmedService.ind {
		}
			OPEN

R55	OPEN	UCA.ind
		&& ArAccessSupported = “True”
		&& (FalApduBody = “UnconfirmedAcknowledgedSend-CommandPDU”
		|| FalApduBody = “UnconfirmedAcknowledgedSend-RequestPDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

#	État courant	Événement ou condition => action	Prochain État
R56	OPEN	Indication de service AR_ASE => ArFspmService NOTE Cet état est fourni pour accéder au FSPM directement à partir de l'utilisateur de la FAL. La fonction ArFspmService génère une primitive FSPM telle que définie ultérieurement dans le présent paragraphe.	OPEN
R58	OPEN	CS.cnf && (FalApduBody = “ConfirmedSend-AffirmativePDU” || FalApduBody = “ConfirmedSend-NegativePDU” || FalApduBody = “ConfirmedSend-ResponsePDU”) && PDU length ≤ NegotiatedMaxPduSizeReceiving && InvokeIdExistent = “True” && SameService = “True” => ConfirmedService.cnf { }, OutstandingServicesRequestingCounter := OutstandingServicesRequestingCounter-1	OPEN

R59	OPEN	CS.cnf
		&& (FalApduBody = “ConfirmedSend-AffirmativePDU”
		|| FalApduBody = “ConfirmedSend-NegativePDU”
		|| FalApduBody = “ConfirmedSend-ResponsePDU”)
		&& PDU length > NegotiatedMaxPduSizeReceiving
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

R60	OPEN	CS.cnf
		&& (FalApduBody = “ConfirmedSend-AffirmativePDU”
		|| FalApduBody = “ConfirmedSend-NegativePDU”
		|| FalApduBody = “ConfirmedSend-ResponsePDU”)
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& InvokeIdExistent = “False”
		=>
		Abort.req {
		AbortIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-responding”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “InvokeID-error-responding”
		},
		
		ResetArep
			CLOSED

7.4 Fonctions

Les tableaux Tableau 5 à Tableau 20 définissent les fonctions internes qui sont utilisées par le diagramme d’états de l'AP-Context.

Tableau 5 – Fonction ResetArep

Nom	ResetArep		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Ferme le Contexte AP AR et initialise tous les éléments de l'attribut de l'AP "*List Of In-Use AR Endpoint Info*" à zéro (0).		

Tableau 6 – Fonction ApContextTest

Nom	ApContextTest		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Localise l'entrée pour l'AREP sélectionné dans l'attribut de l'AP "*List Of In-Use AR Endpoint Info*", vérifie son contenu et assure qu'il y a un AREP correspondant défini pour l'AR_ASE. La façon dont la vérification est effectuée dépend de la mise en œuvre.		

Tableau 7 – Fonction ServicesSupportedTest

Nom	ServicesSupportedTest		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Dès la réception d'une Initiate-RequestPDU, l'AP_ASE appelé doit comparer la Local List (liste locale) des services pris en charge avec ceux contenus dans la PDU Initiate. Le ServicesSupportedTest échoue si la FAL locale ne prend pas en charge tous les services en tant qu'un répondeur que la FAL distante prend en charge en tant qu'un demandeur, ou bien si la FAL distante ne prend pas en charge tous les services en tant qu'un répondeur que la FAL locale prend en charge en tant qu'un demandeur.		

Tableau 8 – Fonction ApExplicitConnection

Nom	ApExplicitConnection		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Se référer à l'AREP pour déterminer si cet AP_ASE prend en charge une connexion explicite entre les AP.		

Tableau 9 – Fonction ImmediateAcknowledge

Nom	ImmediateAcknowledge		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Se référer à l'AREP pour déterminer si cet AP_ASE nécessite un acquittement immédiat dans la couche sous-jacente.		

Tableau 10 – Fonction ConfirmedServiceCheck

Nom	ConfirmedServiceCheck		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Détermine si le service appelé de la FAL est un service confirmé sauf pour les services AR ASE.		

Tableau 11 – Fonction UnconfirmedServiceCheck

Nom	UnconfirmedServiceCheck		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Détermine si le service appelé de la FAL est un service non confirmé sauf pour les services AR ASE.		

Tableau 12 – Fonction ArServiceCheck

Nom	ArServiceCheck		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Détermine si le service appelé est un service AR ASE (service de la FAL ou service de l'AR FSPM).		

Tableau 13 – Fonction ArFspmService

Nom	ArFspmService		
Entrée	**Arep_Id**	**Sortie**	**Service de la FAL ou Service de l'AR FSPM**
Fonction	Générer une primitive de service AR ASE. La relation entre les Services FAL et les Services AR FSPM doit être comme suit: AR-Unconfirmed Send UCS AR-Confirmed Send CS AR-Establish EST AR-Abort Abort		

Tableau 14 – Fonction ArAcceeSupported

Nom	ArAcceeSupported		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Détermine si l'AP accepte les services AR ASE Send (ConfirmedSend ou UnconfirmedSend).		

Tableau 15 – Fonction MaxFalPduLengthTest

Nom	MaxFalPduLengthTest		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Dès la réception d'une Initiate-RequestPDU, l'AP_ASE appelé doit vérifier la longueur maximale demandée de la PDU par rapport à son contexte défini. Les essais suivants sont effectués: 1. Si (ConfiguredMaxPDUSizeSending > MaxPDUSizeReceiving de la PDU Initiate), cet essai échoue; 2. Si (ConfiguredMaxPDUSizeReceiving < MaxPDUSizeSending de la PDU Initiate), cet essai échoue.		

Tableau 16 – Fonction NegotiateOutstandingServices

Nom	NegotiateOutstandingServices		
Entrée	**Arep_Id**	**Sortie**	**True ou False**
Fonction	Dès la réception d'une Initiate-RequestPDU, l'AP_ASE appelé doit effectuer la négociation suivante: 1. Si ConfiguredMaxOustandingServicesRequesting > MaxOutstandingServicesResponding de la PDU Initiate-Request alors NegotiatedMaxOutstandingServicesRequesting = MaxOutstandingServicesResponding de la PDU Initiate-Request sinon NegotiatedMaxOutstandingServicesRequesting = ConfiguredMaxOutstandingServicesRequesting; 2. Si ConfiguredMaxOutstandingServicesResponding < MaxOutstandingServicesRequesting de la PDU Initiate alors NegotiatedMaxOutstandingServicesResponding = ConfiguredMaxOutstandingServicesResponding sinon NegotiatedMaxOutstandingServicesResponding = MaxOutstandingServicesRequesting de la PDU Initiate;		

Tableau 17 – Fonction RequestedServicesSupportedTest

Nom	RequestedServicesSupportedTest		
Entrée	**Arep_Id, Service.primitive**	**Sortie**	**True ou False**
Fonction	Dès la réception d'une demande de service de l'utilisateur de la FAL, l'AP_ASE appelé doit comparer la Local List (liste locale) des services pris en charge avec la primitive de service demandé. Cet essai échoue si la primitive de service n'est pas contenue dans la Local List (liste locale) des services pris en charge.		

Tableau 18 – Fonction IndicatedServicesSupportedTest

Nom	IndicatedServicesSupportedTest		
Entrée	**Arep_Id, service.primitive**	**Sortie**	**True ou False**
Fonction	Dès la réception d'une indication de service de l'AR_ASE, l'AP_ASE appelé doit comparer la Local List (liste locale) des services pris en charge avec la primitive de service indiqué. Cet essai échoue si la primitive de service n'est pas contenue dans la Local List (liste locale) des services pris en charge.		

Tableau 19 – Fonction InvokeIdExistent

Nom	InvokeIdExistent		
Entrée	**Arep_Id, Invoke_Id**	**Sortie**	**True ou False**
Fonction	Dès la réception d'une primitive de service, l'AP_ASE appelé doit 1. comparer la liste locale des invoke ID en cours d'utilisation avec les Invoke ID demandés si la primitive de service est "request"; 2. comparer la liste locale des invoke ID en cours d'utilisation avec les invoke ID répondus si la primitive de service est "response". Cet essai échoue si l'invoke ID n'existe pas dans la liste locale des Invoke ID en cours d'utilisation.		

Tableau 20 – Fonction SameService

Nom	SameService		
Entrée	**Arep_Id, Invoke_Id**	**Sortie**	**True ou False**
Fonction	Dès la réception d'une primitive de service, l'AP_ASE appelé doit 1. comparer la liste locale des services en cours (de réponse) avec le service si la primitive de service est "response"; 2. comparer la liste locale des services en cours (de demande) avec le service si la primitive de service est "confirmation". Cet essai échoue si le service n'est pas le même que dans la liste locale des services outstanding.		

8 FAL service protocol machine (Machine de protocole de service FAL (FSPM))

8.1 Résumé

La "Machine de protocole de service FAL" est commune à tous les types d'AREP. Seules les primitives applicables sont différentes entre les différents types d'AREP. Elle a un état appelé "ACTIVE."

8.2 Définitions des primitives

8.2.1 Primitives échangées entre l'AP-context et le FSPM

Les primitives échangées entre l'AP-Context et le FSPM sont décrites dans le Tableau 21 et le Tableau 22.

Tableau 21 – Primitives émises par l'AP-context vers le FSPM

Nom de primitive	Source	Paramètres associés	Fonctions
EST.req	AP-Context	AREP_ID, Data, Remote_DLCEP_Address	Cette primitive sert à acheminer une primitive "request" du service Establish de l'AP-Context vers le FSPM.
EST.rsp(+)	AP-Context	AREP_ID, Data	Cette primitive sert à acheminer une primitive "response(+)" du service Establish de l'AP-Context vers le FSPM.
EST.rsp(-)	AP-Context	AREP_ID, Data	Cette primitive sert à acheminer une primitive "response(-)" du service Establish de l'AP-Context vers le FSPM.
Abort.req	AP-Context	AREP_ID, Identifier, Reason_Code, Additional_Detail	Cette primitive sert à acheminer une primitive "request" du service Abort de l'AP-Context vers le FSPM.
CS.req	AP-Context	AREP_ID, Data	Cette primitive sert à acheminer une primitive "request" du service Confirmed Send (CS) de l'AP-Context vers le FSPM.
CS.rsp	AP-Context	AREP_ID, Data	Cette primitive sert à acheminer une primitive "response" du service Confirmed Send (CS) de l'AP-Context vers le FSPM.
UCS.req	AP-Context	Arep_Id, Remote_DLSAP_Address, Data	Cette primitive sert à acheminer une primitive "request" du service Unconfirmed Send (UCS) de l'AP-Context vers le FSPM.
UCA.req	AP-Context	AREP_ID, Remote_Dlsap_Address, Data	Cette primitive est utilisée pour envoyer une requête de service non confirmé.

Tableau 22 – Primitives émises par le FSPM vers l'AP-context

Nom de primitive	Source	Paramètres associés	Fonctions
EST.ind	FSPM	AREP_ID, Data	Cette primitive sert à acheminer une primitive "indication" du service Establish du FSPM vers l'AP-Context.
EST.cnf(+)	FSPM	AREP_ID, Data	Cette primitive sert à acheminer une primitive "result(+)" du service Establish du FSPM vers l'AP-Context.
EST.cnf(-)	FSPM	AREP_ID, Data	Cette primitive sert à acheminer une primitive "result(-)" du service Establish du FSPM vers l'AP-Context.
Abort.ind	FSPM	Arep_Id, Locally_Generated, Identifier, Reason_Code, Additional_Detail	Cette primitive sert à acheminer une primitive "indication" du service Abort du FSPM vers l'AP-Context.
CS.ind	FSPM	AREP_ID, Data	Cette primitive sert à acheminer une primitive "indication" du service Confirmed Send (CS) du FSPM vers l'AP-Context.
CS.cnf	FSPM	AREP_ID, Data	Cette primitive sert à acheminer une primitive "confirmation" du service Confirmed Send (CS) du FSPM vers l'AP-Context.
UCS.ind	FSPM	Arep_Id, Remote_DLSAP_Address, Duplicate_FAL-SDU, Data, Local_Timeliness, Remote_Timeliness	Cette primitive sert à acheminer une primitive "indication" du service Unconfirmed Send (UCS) du FSPM vers l'AP-Context.
UCA.ind	FSPM	AREP_ID, Remote_DLSAP_Address, Duplicate_FAL-SDU, Data	Cette primitive sert à acheminer une indication de service non confirmé.

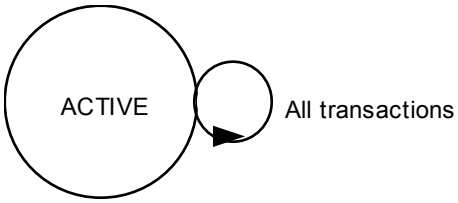
8.2.2 Paramètres des primitives de l'AP-context /du FSPM

Tous les paramètres utilisés dans les primitives échangées entre l'AP-Context et le FSPM sont identiques à ceux définis dans le paragraphe “Operational Services”.

8.3 Table d'états du FSPM

8.3.1 Généralités

Les diagrammes d'états du FSPM sont décrits dans la Figure 25, dans le Tableau 23 et le Tableau 24.



Légende

Anglais	Français
ACTIVE	ACTIF
All transactions	Toutes les transactions

Figure 25 – Diagramme de transitions d'états du FSPM

Tableau 23 – Table d'états du FSPM – transactions de l'émetteur

#	État courant	Événement ou condition => action	Prochain État

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

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

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

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

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

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

#	État courant	Événement ou condition => action	Prochain État

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

S12	ACTIVE	UCA.req
		&& SelectArep (Arep_Id) = "True"
		=>
		UCA_req {
		remote_dlsap_address := Remote_DLSAP_Address,
		user_data := Data
		}
			ACTIVE

S14	ACTIVE	Any FSPM.req && SelectArep (Arep_Id) = “False” => (pas d'actions définies par le protocole, voir NOTE 1 et NOTE 2.)	ACTIVE

NOTE 1 Une primitive générée dans le diagramme d'états de l'émetteur du FSPM est envoyée à une ARPM appropriée qui est sélectionnée par le FSPM en utilisant la fonction SelectArep. Le paramètre Arep_Id fourni par l'AP-Context est l'argument de cette fonction.

NOTE 2 Si la fonction SelectArep retourne la valeur False, il s'agit d'une affaire locale de reporter un tel cas et le FSPM ne génère aucune primitive pour l'ARPM.

Tableau 24 – Table d'états du FSPM – transactions du récepteur

#	État courant	Événement ou condition => action	Prochain État

R1	ACTIVE	EST_ind
		=>
		EST.ind {
		Arep_Id := arep_id,
		data := User_data
		}
			ACTIVE

R2	ACTIVE	EST_cnf(+)
		=>
		EST.cnf(+) {
		Arep_Id := arep_id,
		data := User_data
		}
			ACTIVE

R3	ACTIVE	EST_cnf(-)
		=>
		EST.cnf(-) {
		Arep_Id := arep_id,
		data := User_data
		}
			ACTIVE

R4	ACTIVE	UCS_ind
		=>
		UCS.ind {
		Arep_Id := arep_id,
		Remote_DLSAP_Address := remote_dlsap_address,
		data := user_data,
		}
			ACTIVE

R5	ACTIVE	CS_ind
		=>
		CS.ind {
		Arep_Id := arep_id,
		data := User_data
		}
			ACTIVE

R6	ACTIVE	CS_cnf
		=>
		CS.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

R16	ACTIVE	UCA_ind
		=>
		UCA.ind {
		Arep_Id := arep_id,
		Remote_DLSAP_Address := remote_dlsap_address,
		data := user_data,
		}
			ACTIVE

8.3.2 Fonctions

La fonction utilisée dans ce diagramme d'états est comme présentée dans le Tableau 25.

Tableau 25 – Fonction SelectArep

Nom	**SelectArep**	**Utilisée dans**	**FSPM**
Entrée		Sortie	
Arep_Id		True || False	
Fonction			
Recherche l'entrée de l'AREP qui est spécifié par le paramètre Arep_Id. Le paramètre Arep_Id est fourni avec les primitives de service de l'AP-Context.			

9 Machines de protocole de relations AR (Application relationship protocol machines (ARPM))

9.1 ARPM Queued user-triggered bidirectional-flow control (mis en file d'attente déclenché par l'utilisateur bidirectionnel - contrôle de flux (QUB-FC))

9.1.1 Définitions de la QUB-FCPrimitive

9.1.1.1 Primitives échangées entre l'ARPM et le FSPM

Le Tableau 26 et le Tableau 27 énumèrent les primitives échangées entre l'ARPM et le FSPM.

Tableau 26 – Primitives émises par le FSPM vers l'ARPM

Nom de primitive	Source	Paramètres associés	Fonctions
EST_req	FSPM	User_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "request" du service Establish du FSPM vers l'ARPM.
EST_rsp(+)	FSPM	User_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "response(+)" du service Establish du FSPM vers l'ARPM.
EST_rsp(-)	FSPM	User_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "response(-)" du service Establish du FSPM vers l'ARPM.
Abort_req	FSPM	identifier, reason_code, additional_detail	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "request" du service Abort du FSPM vers l'ARPM.
CS_req	FSPM	User_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "request" du service Confirmed Send (CS) du FSPM vers l'ARPM.
CS_rsp	FSPM	User_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "response" du service Confirmed Send (CS) du FSPM vers l'ARPM.
UCA_req	FSPM	remote_dlsap_address, user_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "request" du service Unconfirmed Acknowledged Send (UCA) du FSPM vers l'ARPM.

Tableau 27 – Primitives émises par l'ARPM vers le FSPM

Nom de primitive	Source	Paramètres associés	Fonctions
EST_ind	ARPM	arep_id, user_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "indication" du service Establish de l'ARPM vers le FSPM.
EST_cnf(+)	ARPM	arep_id, user_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "response(+)" du service Establish de l'ARPM vers le FSPM.
EST_cnf(-)	ARPM	arep_id, user_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "response(-)" du service Establish de l'ARPM vers le FSPM.
Abort_ind	ARPM	arep_id, locally_generated, identifier, reason_code, additional_detail	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive Abort de l'ARPM vers le FSPM.
CS_ind	ARPM	arep_id, user_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "indication" du service Confirmed Send (CS) de l'ARPM vers le FSPM.
CS_cnf	ARPM	arep_id, user_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "confirmation" du service Confirmed Send (CS) de l'ARPM vers le FSPM.
UCA_ind	ARPM	arep_id, remote_dlsap_address, user_data	Il s'agit d'une primitive interne de la FAL utilisée pour acheminer une primitive "indication" du service Unconfirmed Acknowledged Send (UCA) de l'ARPM vers le FSPM.

9.1.2 Paramètres de primitives de FSPM/ARPM

Les paramètres utilisés avec les primitives échangées entre le diagramme FSPM et l'ARPM sont décrits dans le Tableau 28.

Tableau 28 – Paramètres utilisés avec les primitives échangées entre le FSPM et l'ARPM

Nom de paramètre	Description
arep_id	Ce paramètre sert à identifier sans ambiguïté une instance de l'AREP qui a émis une primitive. Un moyen pour une telle identification n'est pas spécifié par la présente norme.
User_data	Ce paramètre achemine les données d'utilisateur de la FAL.
Locally_generated	Ce paramètre achemine la valeur qui est utilisée pour le paramètre Locally_Generated.
IDENTIFIER	Ce paramètre achemine la valeur qui est utilisée pour le paramètre Identifier.
Reason_code	Ce paramètre achemine la valeur qui est utilisée pour le paramètre Reason_Code.
Additional_detail	Ce paramètre achemine la valeur qui est utilisée pour le paramètre Additional_Detail.

9.1.3 Mapping DLL de la Classe QUB-FC AREP

Le présent paragraphe décrit le mapping de la classe QUB-FC AREP vers la couche liaison de données de Type 8 définie dans la CEI 61158-3-8 et la CEI 61158-4-8. Il ne redéfinit pas les attributs DLSAP ou les attributs DLME qui sont ou qui seront définis dans la norme de la couche liaison de données; il définit plutôt la manière dont ils sont utilisés par cette classe AR.

NOTE Le domaine d'application de la présente Norme internationale n'inclut pas de moyen pour configurer et surveiller les valeurs de ces attributs.

Les attributs de Mapping DLL et leurs valeurs permises ainsi que les services DLL utilisés avec la classe QUB-FC AREP sont définis dans le présent paragraphe.

Classe: **Type 8 QubFC**
CLASSE PARENTE: **QueuedUser-TriggeredBidirectional-FlowControlAREP**
ATTRIBUTS:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribut:	RemoteDlcepAddress
3	(m)	Attribut:	QosParameterSet
3.1	(m)	Attribut:	DllPriorityNegotiated
3.2	(m)	Attribut:	MaxConfirmDelay
3.3	(m)	Attribut:	MaxDlsduSizes
3.3.1	(m)	Attribut:	MaxDlsduSizeFromRequestor
3.3.2	(m)	Attribut:	MaxDlsduSizeFromResponder
3.3.3	(m)	Attribut:	MaxDlsduSizeFromRequestorNegotiated
3.3.4	(m)	Attribut:	MaxDlsduSizeFromResponderNegotiated
3.4	(m)	Attribut:	MaxQueueDepth
3.4.1	(m)	Attribut:	MaxSendingQueueDepth
3.4.2	(m)	Attribut:	MaximimReceivingQueueDepth

SERVICES DLL:

1	(m)	OpsService:	DATA de DL

9.1.4 Attributs

9.1.4.1 LocalDlcepAddress

Cet attribut spécifie l'adresse locale du DLCEP et identifie le DLCEP.

La valeur de cet attribut est utilisée comme le paramètre “DLCEP-address” de la DLL.

9.1.4.2 RemoteDlcepAddress

Cet attribut spécifie l'adresse distante du DLCEP et identifie le DLCEP.

9.1.4.3 QosParameterSet

Les attributs QosParameterSet spécifient la qualité de service de la couche liaison de données qui est utilisée par cet AREP. Ces valeurs d'attributs peuvent être négociées avec l'AREP distant.

Cet attribut spécifie la valeur négociée de la priorité de DLL.

9.1.4.4 MaxConfirmDelay

Cet attribut spécifie le délai de confirmation maximal relatif à quelques services de DLL orientés connexion.

9.1.4.5 MaxDlsduSize

MaxDlsduSizeFromRequestor

Cet attribut spécifie la valeur configurée de la longueur maximale d'une PDU de la FAL qui peut être envoyée à partir de l'AREP dont l'attribut Initiator a une valeur “True” vers l'AREP distant.

Cet attribut fournit la valeur relative au paramètre “Maximum DLSDU sizes from responder” de la DLL.

MaxDlsduSizeFromResponder

Cet attribut spécifie la valeur configurée de la longueur maximale d'une PDU de la FAL qui peut être envoyée à partir de l'AREP dont l'attribut Initiator a une valeur “False” vers l'AREP distant.

Cet attribut fournit la valeur relative au paramètre “Maximum DLSDU sizes from responder” de la DLL.

MaxDIsduSizeFromRequestorNegotiated

Cet attribut spécifie la valeur négociée de la longueur maximale d'une PDU de la FAL qui peut être envoyée à partir de l'AREP dont l'attribut Initiator a une valeur “True” vers l'AREP distant.

MaxDIsduSizeFromResponderNegotiated

Cet attribut spécifie la valeur négociée de la longueur maximale d'une PDU de la FAL qui peut être envoyée à partir de l'AREP dont l'attribut Initiator a une valeur “False” vers l'AREP distant.

9.1.4.6 MaxQueueDepth

MaxSendingQueueDepth

Cet attribut spécifie le nombre maximal de PDU de la FAL qui peuvent être mises en file d'attente en vu d'une émission.

Cet attribut fournit aux files d'attente Send (d'envoi) la valeur relative au paramètre “Maximum queue depth” de la DLL.

MaxReceivingQueueDepth

Cet attribut spécifie le nombre maximal de PDU de la FAL qui peuvent être mises en file d'attente à la réception.

Cet attribut fournit aux files d'attente Receive (de réception) la valeur relative au paramètre “Maximum queue depth” de la DLL.

9.1.5 Services DLL

Se référer à la CEI 61158-3-8 pour les descriptions de services de la DLL.

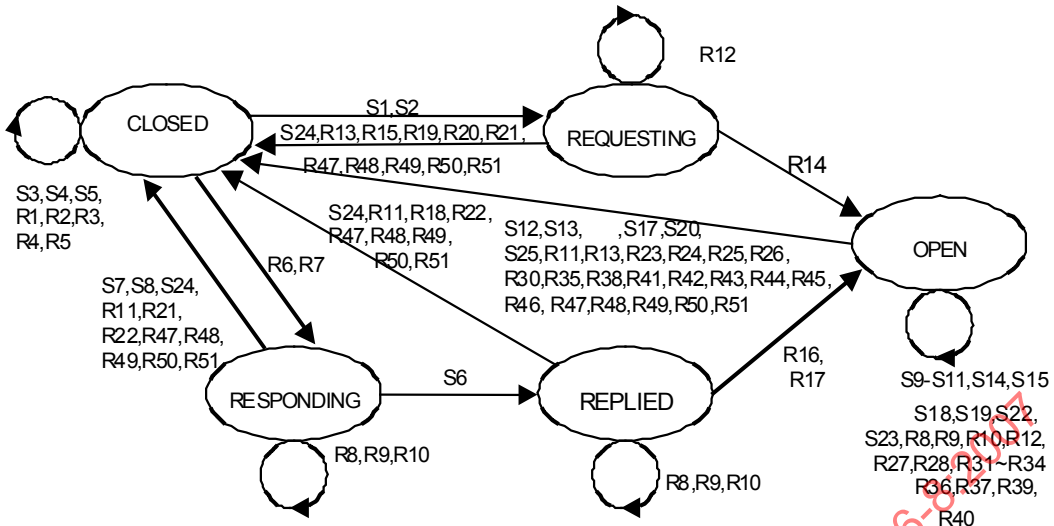
9.1.6 Diagramme d’états de la QUB-FC ARPM

9.1.6.1 États de la QUB-FC ARPM

Les états définis et leurs descriptions de la QUB-FC ARPM sont présentés dans le Tableau 29 et à la Figure 26.

Tableau 29 – États de la QUB-FC ARPM

État	Description
CLOSED	L'AREP est défini, mais il n'est pas capable d'envoyer ou de recevoir des PDU de la FAL. Il peut envoyer ou recevoir des PDU de la FAL du service Establish pendant cet état.
OPEN	L'AREP est défini et il est capable d'envoyer ou de recevoir des PDU de la FAL.
REQUESTING (REQ)	L'AREP a envoyé une PDU de la FAL "Request" du service Establish et attend une réponse de l'AREP distant.
RESPONDING (RSP)	L'AREP a reçu une PDU de la FAL "Request" du service Establish, a délivré une primitive Establish.indication et attend une réponse de son utilisateur.
REPLIED (REPL)	L'AREP Serveur a émis une primitive EST_rsp(+) et attend de recevoir de la DLL une indication "connection-established".
SAME	Indique que le prochain état est similaire à l'état courant.



Légende

Anglais	Français
OPEN	OUVERT
CLOSED	FERMÉ
RESPONDING	RÉPONDANT
REPLIED	RÉPONDU
REQUESTING	DEMANDANT

Figure 26 – Diagramme de transitions d'états de la QUB-FC ARPM

9.1.6.2 Table d'états de la QUB-FC ARPM

Le Tableau 30 et le Tableau 31 définissent le diagramme d'états de la QUB-FC ARPM.

Tableau 30 – Table d'états de la QUB-FC ARPM – transactions de l'émetteur

#	État courant	Événement ou condition => action	Prochain État

S1	CLOSED	EST_req
		&& Initiator = “True”
		&& RemoteAddressConfigurationType := “Free”
		=>
		RemoteDlcepAddress := remote_dlcep_address,
		
		FAL-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 := BuildFAL-PDU (
		fal_pdu_name := “EST_ReqPDU”,
		calling_dlcep_address := LocalDlcepAddress,
		called_dlcep_address := RemoteDlcepAddress,
		fal_data := user_data)
		}
			REQ

S2	CLOSED	EST_req
		&& Initiator = “True”
		&& RemoteAddressConfigurationType := “Linked”
		=>
		FAL-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 := BuildFAL-PDU (
		fal_pdu_name := “EST_ReqPDU”,
		calling_dlcep_address := LocalDlcepAddress,
		called_dlcep_address := RemoteDlcepAddress,
		fal_data := user_data)
		}
			REQ

S3	CLOSED	EST_req
		&& Initiator = “False”
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := “True”,
		Identifier := “FAL”,
		reason_code := “Invalid Event for Role”
		}
			CLOSED

S4	CLOSED	unallowed primitive (primitive non autorisée)
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := “True”,
		Identifier := “FAL”,
		reason_code := “Unallowed AREP primitive in connection establishment“ (primitive AREP non autorisée à l'établissement de connexion)
		}
			CLOSED

S5	CLOSED	Abort_req => IGNORE	CLOSED

S6	Rsp	EST_rsp(+)
		=>
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Connect_rsp”,
		arep_id := GetArepId (),
		responding_address := “default dlsap address”,
		local_dlcep_address := LocalDlcep,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := “EST_RspPDU”,
		fal_data := user_data)
		}
			REPL

#	État courant	Événement ou condition => action	Prochain État

S7	Rsp	EST_rsp(-)
		=>
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := "connection rejection-transient condition" (rejet de connexion - état transistoire),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "EST_ErrPDU",
		fal_data := user_data)
		}
			CLOSED

S8	Rsp	primitive non autorisée
		=>
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		Identifier := "FAL",
		reason_code := "Unallowed AREP primitive in connection establishment" (primitive AREP non autorisée à l'établissement de connexion)
		},
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		reason := "Unallowed AREP primitive in connection establishment" (primitive AREP non autorisée à l'établissement de connexion),
		dlsdu := "null"
		}
			CLOSED

S9	OPEN	CS_req && Role = “Client” || “Peer” && GetCounterValue(OSCC) < maxOSCC && CIU = 0 => FAL-PDU_req { dmpm_service_name := “DMPM_Data_req”, arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := “CS_ReqPDU”, fal_data := user_data) }, IncrementCounter(OSCC)	OPEN
S10	OPEN	CS_req && Role = “Client” || “Peer” && GetCounterValue(OSCC) < maxOSCC && CIU > 0 => FAL-PDU_req { dmpm_service_name := “DMPM_Data_req”, arep_id := GetArepId (), dlsdu := BuildFAL-PDU (fal_pdu_name := “CS_ReqPDU”, fal_data := user_data) }, StartTimer(TC) IncrementCounter(OSCC)	OPEN

S11	OPEN	CS_req
		&& Role = “Client” || “Peer”
		=>
		CS_cnf (-) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			OPEN