

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**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2007 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.

If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de la CEI ou du Comité national de la CEI du pays du demandeur.

Si vous avez des questions sur le copyright de la CEI ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de la CEI de votre pays de résidence.

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It also gives information on projects, replaced and withdrawn publications.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.

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
CODE PRIX

XD

ICS 25.040.40; 35.100.70

ISBN 978-2-83220-636-2

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	6
INTRODUCTION.....	8
1 Scope.....	9
1.1 General.....	9
1.2 Specifications.....	9
1.3 Conformance.....	10
2 Normative references.....	10
3 Terms and definitions.....	10
3.1 ISO/IEC 7498-1 terms.....	10
3.9 ISO/IEC 8822 terms.....	11
3.11 ISO/IEC 9545 terms.....	11
3.16 ISO/IEC 8824 terms.....	11
3.37 ISO/IEC 8825 terms.....	12
3.40 Terms and definitions from IEC 61158-5-8.....	12
3.47 Other terms and definitions.....	12
4 FAL syntax description.....	13
4.1 FAL-AR PDU abstract syntax.....	13
4.2 Abstract syntax of PDUBody.....	16
4.3 Type definitions for ASEs.....	21
4.4 Object definitions.....	24
4.5 Abstract syntax of Data types.....	26
5 Transfer syntax.....	27
5.1 Peripherals encoding rules (PER).....	27
5.2 Encoding of APDU types.....	27
5.3 Encoding of tagged type values.....	29
5.4 Encoding of simple values.....	30
6 Protocol machine overview.....	36
7 AP-context state machine.....	36
7.1 Primitive definitions.....	36
7.2 State machine description.....	37
7.3 AP to AP-context initiation state transitions.....	38
7.4 Functions.....	49
8 FAL service protocol machine (FSPM).....	52
8.1 Summary.....	52
8.2 Primitive definitions.....	52
8.3 FSPM state tables.....	54
9 Application relationship protocol machines (ARPMs).....	57
9.1 Queued user-triggered bidirectional-flow control (QUB-FC) ARPM.....	57
9.2 Buffered network-scheduled unidirectional (BNU) ARPM.....	79
9.3 Queued user-triggered bidirectional – transparent mode (QUB-TM) ARPM.....	87
10 DLL mapping protocol machine.....	91
10.1 Overview.....	91
10.2 Primitive definitions.....	92
10.3 DMPM state machine.....	95
Bibliography.....	

Figure 1 – APDU overview	27
Figure 2 – APDU header	27
Figure 3 – PDU with type extension	27
Figure 4 – PDU with address extension	28
Figure 5 – PDU with type and length extension	28
Figure 6 – Example of an Establish-Request PDU	28
Figure 7 – Encoding of a PRIVATE tagged value	29
Figure 8 – Encoding of a context specific tagged value	29
Figure 9 – Identification information fields	29
Figure 10 – ID-info for tag 0 .. 14 , length entry 0 .. 6	30
Figure 11 – ID-info for tag 15 .. 255 , length entry 0 .. 6	30
Figure 12 – ID-info for tag 0 .. 14 , length entry 7 .. 255	30
Figure 13 – ID-info for tag 15 .. 255 , length entry 7 .. 255	30
Figure 14 – Encoding of Boolean value TRUE	30
Figure 15 – Encoding of Boolean value FALSE	30
Figure 16 – Encoding of Strings	31
Figure 17 – Encoding of BinaryDate value	32
Figure 18 – Encoding of BinaryDate2000 value	32
Figure 19 – Encoding of Time-of-day value	33
Figure 20 – Encoding of Time-difference value	33
Figure 21 – Encoding of Time value	34
Figure 22 – Example for an Object definition	35
Figure 23 – Primitives exchanged between protocol machines	36
Figure 24 – AP to AP-context initiation state machine	38
Figure 25 – State transition diagram of FSPM	54
Figure 26 – State transition diagram of QUB-FC ARPM	60
Figure 27 – State transition diagram of the BNU ARPM	82
Figure 28 – State transition diagram of QUB-TM AREP	89
Figure 29 – State transition diagram of DMPM	95
Table 1 – Primitives issued by FAL-user to AP-context	36
Table 2 – Primitives issued by AP-context to FAL-user	37
Table 3 – AP-context state machine sender transactions	38
Table 4 – AP-context state machine receiver transactions	42
Table 5 – Function ResetArep	50
Table 6 – Function ApContextTest	50
Table 7 – Function ServicesSupportedTest	50
Table 8 – Function ApExplicitConnection	50
Table 9 – Function ImmediateAcknowledge	50
Table 10 – Function ConfirmedServiceCheck	50
Table 11 – Function UnconfirmedServiceCheck	50
Table 12 – Function ArServiceCheck	51

Table 13 – Function ArFspmService	51
Table 14 – Function ArAcceeSupported	51
Table 15 – Function MaxFalPduLengthTest	51
Table 16 – Function NegotiateOutstandingServices	51
Table 17 – Function RequestedServicesSupportedTest	52
Table 18 – Function IndicatedServicesSupportedTest	52
Table 19 – Function InvokeldExistent	52
Table 20 – Function SameService.....	52
Table 21 – Primitives issued by AP-context to FSPM	53
Table 22 – Primitives issued by FSPM to AP-context	53
Table 23 – FSPM state table – sender transactions	54
Table 24 – FSPM state table – receiver transactions	56
Table 25 – Function SelectArep	56
Table 26 – Primitives issued by FSPM to ARPM	57
Table 27 – Primitives issued by ARPM to FSPM	57
Table 28 – Parameters used with primitives exchanged between FSPM and ARPM	58
Table 29 – QUB-FC ARPM states	60
Table 30 – QUB-FC ARPM state table – sender transactions	61
Table 31 – QUB-FC ARPM state table – receiver transactions	66
Table 32 – Function GetArepId ()	77
Table 33 – Function BuildFAL-PDU	77
Table 34 – Function FAL_Pdu_Type	78
Table 35 – Function AREPContextCheck().....	78
Table 36 – Function AbortIdentifier	78
Table 37 – Function AbortReason	78
Table 38 – Function AbortDetail.....	78
Table 39 – Function StartTimer.....	79
Table 40 – Function StopTimer	79
Table 41 – Function ResetCounters	79
Table 42 – Function IncrementCounter	79
Table 43 – Function DecrementCounter	79
Table 44 – Function GetCounterValue	79
Table 45 – Primitives issued by FSPM to ARPM	80
Table 46 – Primitives issued by ARPM to FSPM	80
Table 47 – Parameters used with primitives exchanged between FSPM and ARPM	80
Table 48 – BNU ARPM states	82
Table 49 – BNU ARPM state table – sender transactions	82
Table 50 – BNU ARPM state table – receiver transactions	83
Table 51 – Function GetArepId ()	86
Table 52 – Function BuildFAL-PDU	86
Table 53 – Function FAL_Pdu_Type	86
Table 54 – Function AbortIdentifier	86
Table 55 – Function AbortReason	86

Table 56 – Function AbortDetail.....	86
Table 57 – Primitives issued by FAL to ARPM	87
Table 58 – Primitives issued by ARPM to FAL	87
Table 59 – Parameters used with primitives exchanged between FAL and ARPM	87
Table 60 – QUB-TM ARPM states	89
Table 61 – QUB-TM state table - sender transactions	89
Table 62 – QUB-TM state table - receiver transactions	90
Table 63 – Function GetArepld ()	90
Table 64 – Function BuildFAL-PDU.....	90
Table 65 – Function FAL_Pdu_Type	90
Table 66 – Function ResetCounters	91
Table 67 – Function IncrementCounter	91
Table 68 – Function DecrementCounter	91
Table 69 – Function GetCounterValue	91
Table 70 – Primitives issued by ARPM to DMPM	92
Table 71 – Primitives issued by DMPM to ARPM	93
Table 72 – Parameters used with primitives exchanged between ARPM and DMPM	94
Table 73 – Primitives exchanged between data-link layer and DMPM	94
Table 74 – DMPM state descriptions.....	95
Table 75 – DMPM state table – sender transactions	95
Table 76 – DMPM state table – receiver transactions.....	99
Table 77 – Function PickArep	100
Table 78 – Function FindAREP	100
Table 79 – Function SelectNextArep	100
Table 80 – Function ArepRole.....	101
Table 81 – Function FalArHeader	101
Table 82 – Function AddUcsPduHeader.....	101
Table 83 – Function RemoveUcsPduHeader	101
Table 84 – Function DILinkStatus	101

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  
    optionalParametersMap  
    additionalCode  
    additionalDescription  
}
```

[0] IMPLICIT ErrorClass,
[10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
[1] IMPLICIT Integer16 OPTIONAL,
[2] IMPLICIT VisibleString OPTIONAL,

4.2.6 Error Fi type

```
ErrorFiType ::= SEQUENCE {  
    errorClass  
    additionalCode  
    fiState  
}
```

[0] IMPLICIT ErrorClass,
[1] IMPLICIT Integer16 OPTIONAL,
[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-unsupported (10)
    },
    objectDescription  [7] IMPLICIT Integer8 {
        other (0),
        name-length-overflow (1),
        od-overflow (2),
        od-write-protected (3),
        extension-length-overflow (4),
        od-description-length-overflow (5),
        operational-problem (6)
    },
    conclude           [9] IMPLICIT Integer8 {
        other (0),
        further-communication-required (1)
    },
    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 {
    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)
}
-- The Password (Unsigned8) is encoded as a bit string.

```

4.3.2 AR ASE types

4.3.2.1 Reason code

```
ReasonCode ::= Unsigned8
```

4.3.2.1.1 Additional detail

```
AdditionalDetail ::= 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  $-2^7 \leq i \leq 2^7-1$ 
Integer16 ::= INTEGER (-32768..+32767) -- range  $-2^{15} \leq i \leq 2^{15}-1$ 
Integer32 ::= INTEGER -- range  $-2^{31} \leq i \leq 2^{31}-1$ 
```

4.5.3 Notation for the Unsigned type

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

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.

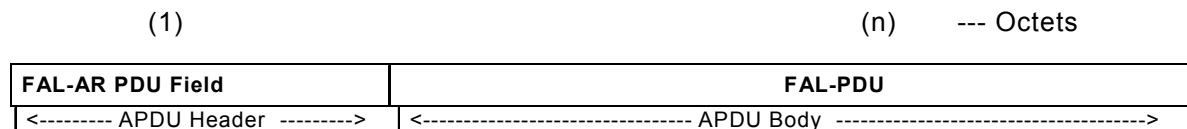


Figure 1 – APDU overview

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.

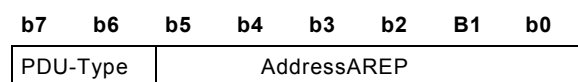


Figure 2 – APDU header

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.

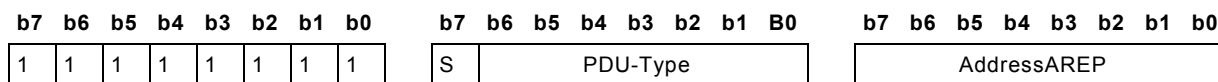


Figure 5 – PDU with type and length extension

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-triggered) 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

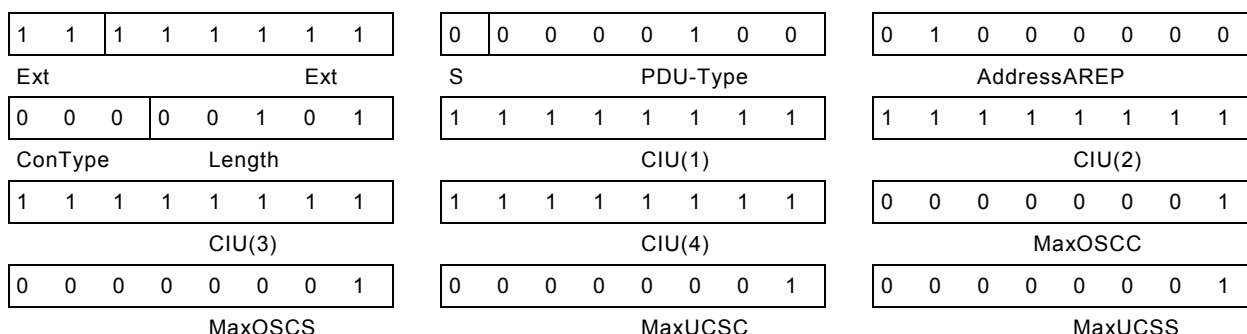


Figure 6 – Example of an Establish-Request PDU

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.

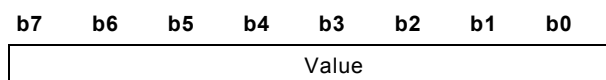


Figure 7 – Encoding of a PRIVATE tagged value

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 or 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).

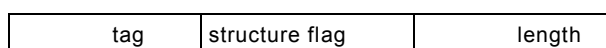


Figure 9 – Identification information fields

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).

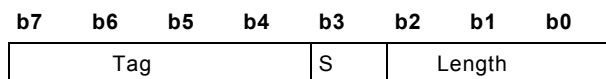


Figure 10 – ID-info for tag 0 .. 14 , length entry 0 .. 6

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).

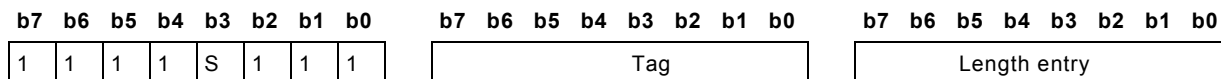


Figure 13 – ID-info for tag 15 .. 255 , length entry 7 .. 255

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

- 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.
- 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

- 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.
- 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

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

- The encoding of a BinaryDate value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the BinaryDate value.
- 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	0...59 999 ms
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
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

- The encoding of a BinaryDate2000 value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the BinaryDate2000 value.
- 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	0...59 999 ms
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
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

- The encoding of a Time Of Day value shall be primitive.
- 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 milliseconds since midnight
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
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 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 milliseconds
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
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	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}	
1									
2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}	
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}	
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}	
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	signed integer
8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	of 8 octet length of 1/32ms unit
MSB									

Figure 21 – Encoding of Time value

5.4.11 Encoding of a Null value

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

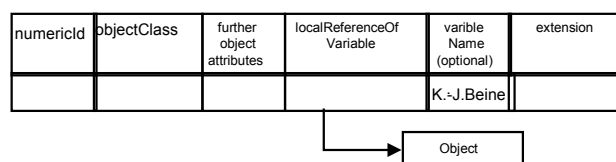


Figure 22 – Example for an Object definition

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.

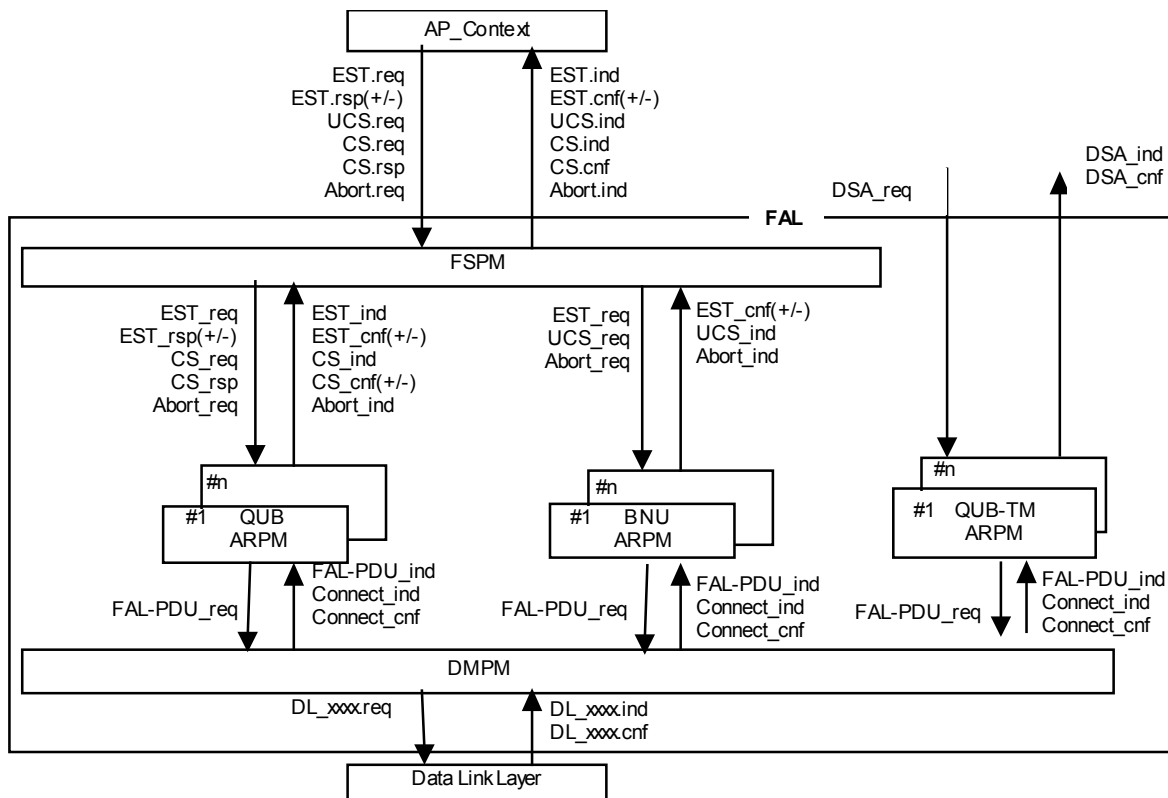


Figure 23 – Primitives exchanged between protocol machines

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"

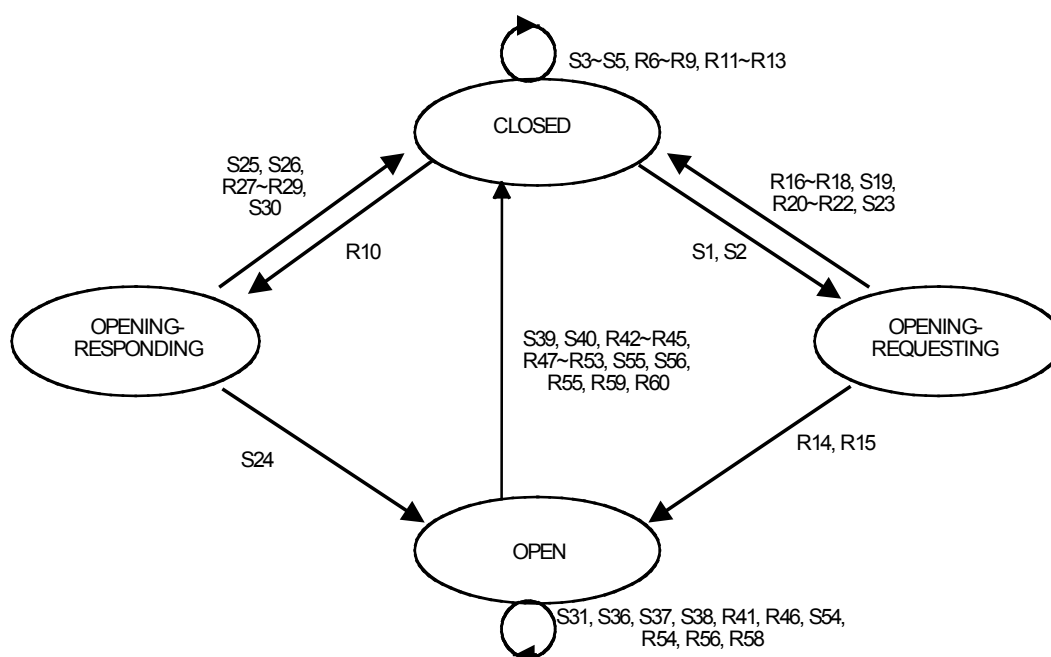


Figure 24 – AP to AP-context initiation state machine

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 “FalApuBody” 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 { FalApuBody := “Establish-CommandPDU” } or EST.req { FalApuBody := “Establish-RequestPDU” } or EST.req { FalApuBody := “Establish-Request2PDU” }	REQ
S2	CLOSED	Initiate.req && ApContextTest = “True” && ApExplicitConnection = “False” => EST.req { FalApuBody := “NULL” }	REQ
S3	CLOSED	Initiate.req && ApContextTest = “False” => Terminate.ind { LocallyGenerated := “True”, TerminatIdentifier := “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", Terminateldentifier := "FAL", ReasonCode := "User error" }	CLOSED
S19	REQ	Terminate.req => Abort.req { Originator := "Terminateldentifier 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", Terminateldentifier := "FAL", ReasonCode := "User error" }, ResetArep	CLOSED
S24	RSP	Initiate.rsp(+) => EST.rsp(+) { FalApuBody := "Establish-AffirmativePDU" } or EST.rsp(+) { FalApuBody = "Establsih-ResponsePDU" }	OPEN
S25	RSP	Initiate.rsp(-) => EST.rsp(-){ FalApuBody := "Establish-NegativePDU" }, or EST.rsp(-) { FalApuBody = "Establsih-ErrorPDU" }, ResetArep	CLOSED
S26	RSP	Terminate.req => Abort.req { Originator := "Terminateldentifier 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", Terminateldentifier := "FAL", ReasonCode := "User error" }, ResetArep	CLOSED
S31	OPEN	ConfirmedService.req && ConfirmedServiceCheck = "True" && OutstandingServicesRequestingCounter < NegotiatedMaxOutstanding- ServicesRequesting && InvokeldExistent = "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 := "Terminateldentifier of Terminate.req", ReasonCode := "ReasonCode of Terminate.req" }, ResetArep	CLOSED
S40	OPEN	Faulty, unknown or not-allowed FAL service.primitive => Abort.req { Abortldentifier := "FAL", ReasonCode := "User error" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "User error" }, ResetArep	CLOSED
S54	OPEN	ConfirmedService.rsp && ConfirmedServiceCheck = "True" && InvokeldExistent = "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 determine whether ConfirmedSend-AffirmativePDU or ConfirmedSend-NegativePDU is used.	OPEN
S55	OPEN	ConfirmedService.rsp && ConfirmedServiceCheck = "True" && InvokeldExistent = "False" => Abort.req { Originator := "FAL", ReasonCode := "Invokeld-error-response" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "Invokeld-error-response" }, ResetArep	CLOSED

#	Current state	Event or condition => action	Next State
S56	OPEN	ConfirmedService.rsp && ConfirmedServiceCheck = "True" && InvokeldExistent = "True" && SameService = "False" => Abort.req { Originator := "FAL", ReasonCode := "Service-error" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "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 && (FalAduBody = "Establish-CommandPDU" FalAduBody = "Establish-RequestPDU" FalAduBody = "Establish-Request2PDU") && ApContextTest = "True" && MaxFalPduLengthTest = "False" => EST.rsp(-) { FalAduBody := "Establish-NegativePDU", ErrorCode := "Max-Fal-Fdu-Size-Insufficient" } or EST.rsp(-) { FalAduBody := "Establish-ErrorPDU", ErrorCode := "Max-Fal-Fdu-Size-Insufficient" }	CLOSED
R13	CLOSED	EST.ind && (FalAduBody = "Establish-CommandPDU" FalAduBody = "Establish-RequestPDU" FalAduBody = "Establish-Request2PDU") && ApContextTest = "True" && MaxFalPduLengthTest = "True" && ServicesSupportedTest = "False" => EST.rsp(-) { FalAduBody := "Establish-NegativePDU", ErrorCode := "Service-Not-Supported" } or EST.rsp(-) { FalAduBody := "Establish-ErrorPDU", ErrorCode := "Service-Not-Supported" }	CLOSED
R14	REQ	EST.cnf(+) && (FalAduBody = "Establish-AffirmativePDU" FalAduBody = "Establish-ResponsePDU") && ApExplicitConnection = "True" => Initiate.cnf(+) {	OPEN
R15	REQ	EST.cnf(+) && FalAduBody = "NULL" && ApExplicitConnection = "False" => Initiate.cnf(+) {	OPEN
R16	REQ	EST.cnf(-) && (FalAduBody = "Establish-NegativePDU" FalAduBody = "Establish-ErrorPDU") && ApExplicitConnection = "True" => "Initiate.cnf(-) { ErrorCode := "Errorcode of EST.cnf(-)" }, ResetArep	CLOSED
R17	REQ	EST.cnf(-) && FalAduBody = "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", Terminateldentifier := "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", Terminateldentifier := "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", Terminateldentifier := "FAL", ReasonCode := "Connection State Conflict" }, ResetArep	CLOSED
R22	REQ	EST.cnf && FalApduBody = " not allowed, unknown or faulty FAL PDU" => Abort.req { Abortldentifier := "FAL", ReasonCode := "FAL PDU error" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "FAL PDU error" }, ResetArep	CLOSED
R27	RSP	Abort.ind => Terminate.ind { LocallyGenerated := "LocallyGenerated of Abort.ind", Terminateldentifier := "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", TerminatIdentifier := "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", TerminatIdentifier := "AP_ASE", ReasonCode := "Connection State Conflict with AR_ASE" }, ResetArep	CLOSED
R41	OPEN	CS.ind && (FalApduBody = "ConfirmedSend-CommandPDU" FalApduBody = "ConfirmedSend-RequestPDU") && PDU length ≤ NegotiatedMaxPduSizeReceiving && OutstandingServicesRespondingCounter < NegotiatedMaxOutstanding- ServicesResponding && InvokeldExistent = "False" && IndicatedServiceSupportedTest = "True" => ConfirmedService.ind { }, OutstandingServicesRespondingCounter := OutstandingServicesRespondingCounter+1	OPEN
R42	OPEN	CS.ind && (FalApduBody = "ConfirmedSend-CommandPDU" FalApduBody = "ConfirmedSend-RequestPDU") && PDU length > NegotiatedMaxPduSizeReceiving => Abort.req { Originator := "FAL", ReasonCode := "PDU-size" }, Terminate.ind { LocallyGenerated := "True", TerminatIdentifier := "FAL", ReasonCode := "PDU-size" }, ResetArep	CLOSED

#	Current state	Event or condition => action	Next state
R43	OPEN	CS.ind && (FalApduBody = "ConfirmedSend-CommandPDU" FalApduBody = "ConfirmedSend-RequestPDU") && PDU length ≤ NegotiatedMaxPduSizeReceiving && OutstandingServicesResponding ≥ NegotiatedMaxOutstanding- ServicesResponding => Abort.req { Originator := "FAL", ReasonCode := "Max-services-overflow" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "Max-Services-Overflow" }, ResetArep	CLOSED
R44	OPEN	CS.ind && (FalApduBody = "ConfirmedSend-CommandPDU" FalApduBody = "ConfirmedSend-RequestPDU") && PDU length ≤ NegotiatedMaxPduSizeReceiving && OutstandingServicesRespondingCounter < NegotiatedMaxOutstanding- ServicesResponding && InvokeldExistent = "True" => Abort.req { Originator := "FAL", ReasonCode := "Invokeld-error-request" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "Invokeld-error-request" }, ResetArep	CLOSED
R45	OPEN	CS.ind && (FalApduBody = "ConfirmedSend-CommandPDU" FalApduBody = "ConfirmedSend-RequestPDU") && PDU length ≤ NegotiatedMaxPduSizeReceiving && OutstandingServicesRespondingCounter < NegotiatedMaxOutstandingServicesResponding && InvokeldExistent = "False" && IndicatedServiceSupportedTest = "False" => Abort.req { Originator := "FAL", ReasonCode := "Feature-not-supported" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "Feature-not-supported" }, ResetArep	CLOSED
R46	OPEN	UCS.ind && (FalApduBody = "UnconfirmedSend-CommandPDU" FalApduBody = "UnconfirmedSend-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 = "UnconfirmedSend-PDU") && PDU length > NegotiatedMaxPduSizeReceiving => Abort.req { Originator := "FAL", ReasonCode := "PDU-size" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "PDU-size" }, ResetArep	CLOSED
R48	OPEN	UCS.ind && (FalApduBody = "UnconfirmedSend-CommandPDU" FalApduBody = "UnconfirmedSend-PDU") && PDU length ≤ NegotiatedMaxPduSizeReceiving && IndicatedServiceSupportedTest = "False" => Abort.req { Originator := "FAL", ReasonCode := "Feature-not-supported" }, Terminate.ind { LocallyGenerated := "True", Terminateldentifier := "FAL", ReasonCode := "Feature-not-supported" }, ResetArep	CLOSED
R49	OPEN	Abort.ind => Terminate.ind { LocallyGenerated := "LocallyGenerated of Abort.ind", Terminateldentifier := "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", Terminateldentifier := "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", Terminateldentifier := "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", Terminateldentifier := "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", Terminateldentifier := "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", Terminateldentifier := "FAL", ReasonCode := "PDU-size" }, ResetArep	CLOSED