TECHNICAL SPECIFICATION

IEC 61158-6

First edition
1999-03

Digital data communications for measurement and control — Fieldbus for use in industrial control systems

Part 6: Application Layer protocol specification

Reference number
IEC 61158-6:1999(E)

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
  Published yearly with regular updates
  (On-line catalogue)*
- **IEC Bulletin**
  Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams.*

* See web site address on title page.

TECHNICAL SPECIFICATION

IEC 61158-6

First edition
1999-03

Digital data communications for measurement and control — Fieldbus for use in industrial control systems

Part 6: Application Layer protocol specification

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http://www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XH

For price, see current catalogue

Contents

Page

Page

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL – FIELDBUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –

Part 6: Application Layer protocol specification

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the 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, the IEC publishes International Standards. 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. The 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 the 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 National Committees.

3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.

4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.

5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.

6) Attention is drawn to the possibility that some of the elements of this technical specification may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

The main task of IEC technical committees is to prepare International Standards. In exceptional circumstances, a technical committee may propose the publication of a technical specification when

- the required support cannot be obtained for the publication of an International Standard, despite repeated efforts, or
- the subject is still under technical development or where, for any other reason, there is the future but no immediate possibility of an agreement on an International Standard.

Technical specifications are subject to review within three years of publication to decide whether they can be transformed into International Standards.

IEC 61158-6, which is a technical specification, has been prepared by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

The text of this technical specification is based on the following documents:

Enquiry draft	Report on voting
65C/200/FDIS	65C/208+208A/RVD

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

This publication has been drafted in accordance with the ISO/IEC Directives, Part 3.

IEC 61158 consists of the following parts, under the general title *Digital data communications for measurement and control — Fieldbus for use in industrial control systems*:

Part 1:	Introductory guide (under preparation)
Part 2:	Physical layer specification and service definition
Part 3:	Data Link Service definition
Part 4:	Data Link Protocol specification
Part 5:	Application Layer service definition
Part 6:	Application Layer protocol specification
Part 7:	System management (under consideration)
Part 8:	Conformance testing (under consideration)

Annexes A to O form an integral part of this technical specification.

Annexes P to R are for information only.

This publication will be reviewed by the committee responsible for its preparation before 2002. Information relating to confirmation, amendment or revision of the publication is available from the IEC web site (http:/www.iec.ch) or from IEC Central Office.

A bilingual version of this technical specification may be issued at a later date.

INTRODUCTION

This technical specification describes the Fieldbus Application Layer (FAL) protocol that defines the information interchange and the interactions between Application Entity invocations (AE-Is) to support the services defined in IEC 61158-5.

An Application Process uses the Fieldbus Application Layer services to exchange information with other Application Processes. The services define the abstract interface between the application process and the Application Layer.

The Application Layer protocol is the set of rules that governs the format and meaning of the information exchange between the Application Layers in various devices. The Application Layer uses the protocol to implement the Application Layer services definitions.

The protocol machine defines the various states of an Application Layer and the valid transitions between the states. It may be considered as a finite state machine. The protocol machine is described using state tables. The information is exchanged between the application process and the protocol machine through application service data units. The protocol machine exchanges information with other protocol machines through application protocol data units (APDU).

This set of Application Layer standards does not specify individual implementations or products, nor does it constrain the implementations of Application Entities (AEs) and interfaces within the industrial automation system.

This set of Application Layer standards does not contain test procedures to ensure compliance with such requirements.

DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL – FIELDBUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –

Part 6: Application Layer protocol specification

1 Scope

The Fieldbus Application Layer (FAL) is an Application Layer communication standard designed to support the conveyance of time-critical application requests and responses among devices in an automation environment. 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 technical specification specifies interactions between remote applications in terms of

- the encoding rules that are applied to all the Application Layer Protocol Data Units (APDUs);
- the formal Abstract Syntax definitions of such APDUs;
- the protocol state machine descriptions that handle the APDUs and the primitives in the correct sequences;
- the mappings of the APDUs to and from the Data Link Layer services defined in IEC 61158-3.

The FAL encoding rules are designed assuming that both the encoder (sender) and the decoder (receiver) have the common knowledge of the abstract syntax. Wherever possible, data types identifiers are not encoded and transferred over the network.

NOTE This is why the Abstract Syntax Notation One / Basic Encoding Rule is not practical for the FAL.

The purpose of this part of this technical specification is to define the protocol provided

a) to the Fieldbus Data Link Layer at the boundary between the Application and Data Link Layers of the Fieldbus Reference Model, and

b) to the System Management at the boundary between the System Management and Application Layers of the Fieldbus Reference Model.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61158. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61158 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards

IEC 61158-3:1999, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 3: Data Link service definitions*

IEC 61158-4:1999, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 4: Data Link protocol specifications*

IEC 61158-5:1999, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 5: Application Layer service definitions*

ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

ISO/IEC 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*

ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure*

ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

3 Definitions

For the purpose of this technical specification the following definitions apply.

3.1 Definitions from other ISO/IEC Standards

3.1.1 Definitions from ISO/IEC 7498-1

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

3.1.2 Definitions from ISO/IEC 8822

a) abstract syntax;
b) presentation context.

3.1.3 Definitions from ISO/IEC 9545

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

3.1.4 Definitions from ISO/IEC 8824

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

3.1.5 Definitions from ISO/IEC 8825

a) encoding (of a data value);
b) data value;
c) Identifier Octets (the singular form is used in this technical specification);
d) Length Octet(s) (both singular and plural forms are used in this technical specification);
e) Contents Octets.

3.2 Definitions from IEC 61158-5

a) application relationship;
b) conveyance path;
c) client;
d) dedicated AR;
e) dynamic AR;
f) error class;
g) error code;
h) name;
i) numeric identifier;
j) peer;
k) pre-defined AR endpoint;
l) pre-established AR endpoint;
m) publisher;
n) server.

3.3 Other definitions

The following definititions are used in this technical specification.

3.3.1 called

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

3.3.2 calling

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

3.3.3 interoperability

capability of User Layer entities to perform coordinated and cooperative operations using the services of the FAL

3.3.4 management information

network accessible information that supports the management of the Fieldbus environment

3.3.5 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

3.3.6 resource

resource is a processing or information capability of a subsystem

3.3.7 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

3.4 Abbreviations and symbols

AE	Application Entity
AE-I	Application Entity Invocation
AL	Application Layer
AP	Application Process
Ap_	Prefix for Data types defined for AP ASE
Ar_	Prefix for Data types defined for AR ASE
APDU	Application Protocol Data Unit
AR	Application Relationship
AREP	Application Relationship End Point
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
BCD	Binary Coded Decimal
BER	Basic Encoding Rule
cnf	confirmation primitive
Dl_	Prefix for Data types defined for Data Link Layer types
DL	Data Link
DLC	Data Link Connection

DLCEP	Data Link Connection End Point
DLSAP	Data Link Service Access Point
DLSDU	Data Link Service Data Unit
Dt_	Prefix for Data types defined for Data Type ASE
Err	Error (used to indicate an APDU type)
Er_	Prefix for Error types defined
Ev_	Prefix for Data types defined for Event ASE
FAL	Fieldbus Application Layer
Fi_	Prefix for Data types defined for Function Invocation ASE
FIFO	First In First Out
Gn_	Prefix for Data Types defined for general use
ID	Identifier
IEC	International Electrotechnical Commission
in	input primitive
ind	indication primitive
ISA	Instrument Society of America
ISO	International Organization for Standardization
LAS	Link Active Scheduler
Lr_	Prefix for Data types defined for Load Region ASE
lsb	least significant bit
Mn_	Prefix for Data Types defined for Management ASE
msb	most significant bit
out	output primitive
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
QoS	Quality Of Service
Req	Request (used to indicate an APDU type)
req	request primitive
Rsp	Response (used to indicate an APDU type)
rsp	response primitive
SAP	Service Access Point
SDU	Service Data Unit
ToS	Type Of Service
Vr_	Prefix for Data types defined for Variable ASE

3.5 Conventions

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

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in the ISA-S50.02-5 specification. The service specification defines the services that are provided by the ASE.

3.5.1 General Conventions

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

3.5.2 Conventions for Class Definitions

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

3.5.3 Abstract Syntax Conventions

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

3.6 Conventions used in State Machines

The State Machines are described as follows:

Table 1 – Conventions used for State Machines

No.	Current State	Events Actions	Next state
Name of this transition.	The current state to which this state transition applies.	Events or conditions that trigger this state transaction. The actions that are taken when the above events or conditions are met. The actions are always indented below events or conditions.	The next state after the actions in this transition is taken.

The conventions used in the state machines are as follows:

:=	Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.
xxx	A parameter name. Example: Identifier := reason means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'
"xxx"	Indicates fixed value. Example: Identifier := "abc" means value "abc" is assigned to a parameter named 'Identifier.'
=	A logical condition to indicate an item on the left is equal to an item on the right.
<	A logical condition to indicate an item on the left is less than the item on the right.
>	A logical condition to indicate an item on the left is greater than the item on the right.
<>	A logical condition to indicate an item on the left is not equal to an item on the right.
&&	Logical "AND"
||	Logical "OR"

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

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

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

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

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

4 FAL Syntax Description

FalArHeader

```
FalArHeader ::= Unsigned8 {
      -- bit 8          FAL Protocol Specifier       (Always 0 for the protocol defined by this part of ISA-S50.02)
      -- bit 7 - 4      Protocol Identifier          (Identifies abstract syntax, revision, and encoding rules)
      -- bit 3          Protocol Specific bit        (Reserved for each protocol to use)
      -- bit 2-1        PDU Identifier               (Identifies a PDU type within a Protocol Identifier)
}
```

The FalArHeader shall be the first octet of all the FAL PDUs. It enables multiple protocols as well as encoding rules to coexist on the same network. “Annex-A FAL Header” defines the code points for the currently identified combinations of protocol and encoding rules.

4.1 FAL-AR PDU Abstract Syntax 1

The productions defined here shall be used with the Compact Encoding Rules, and it is strongly recommended that they be used with the messaging and buffered encoding rules to produce the shortest PDUs that are suitable for a limited bandwidth underlying layer.

```
FalArPDU ::=     ConfirmedSend-CommandPDU
                 || ConfirmedSend-AffirmativePDU
                 || ConfirmedSend-NegativePDU
                 || UnconfirmedSend-CommandPDU
                 || UnconfirmedAcknowledgedSend-CommandPDU
                 || UnconfirmedAcknowledgedSend-AffirmativePDU
                 || Idle-CommandPDU
                 || AR-XON-OFF-CommandPDU
                 || Establish-CommandPDU
                 || Establish-AffirmativePDU
                 || Establish-NegativePDU
                 || Abort-PDU
```

4.1.1 Confirmed Send Service

```
ConfirmedSend-CommandPDU ::= SEQUENCE {
      FalArHeader,
      InvokeID,
      ConfirmedServiceRequest
}

ConfirmedSend-AffirmativePDU ::= SEQUENCE {
      FalArHeader,
      InvokeID,
      ConfirmedServiceResponse
}

ConfirmedSend-NegativePDU ::= SEQUENCE {
      FalArHeader,
      InvokeID,
      ConfirmedServiceError
}
```

4.1.2 Unconfirmed Send Service

```
UnconfirmedSend-CommandPDU ::= SEQUENCE {
      FalArHeader,
      InvokeID,
      UnconfirmedServiceRequest
}
```

4.1.3 Unconfirmed Acknowledged Send Service

```
UnconfirmedAcknowledgedSend-CommandPDU ::= SEQUENCE {
      FalArHeader,
      unconfirmed-PDU                          [4] IMPLICIT SEQUENCE {
         InvokeID,
         UnconfirmedServiceRequest
}
```

```
UnconfirmedAcknowledgedSend-AffirmativePDU::= {
      FalArHeader
}
```

4.1.4 Idle Send Service

```
IdleSend-CommandPDU::= {
      FalArHeader
}
```

4.1.5 AR-XON-OFF Send Service

```
AR-XON-OFF-CommandPDU ::= SEQUENCE {
      FalArHeader,
      AR-XON-OFFPDU                          [7] IMPLICIT SEQUENCE {
           Invoke ID,
           XON-OFF
      }
}
```

4.1.6 Establish Service

```
Establish-CommandPDU ::= SEQUENCE {
      FalArHeader,
      RespondingAREP,
      RequestingAREP,
      MAXOSCC,
      MAXOSCS,
      MAXUCSC,
      MAXUCSS,
      CIU,
      InvokeID,
      Initiate-RequestPDU
}

Establish-AffirmativePDU ::= SEQUENCE {
      FalArHeader,
      InvokeID,
      Initiate-ResponsePDU
}

Establish-NegativePDU ::= SEQUENCE {
      FalArHeader,
      InvokeID,
      Initiate-ErrorPDU
}
```

4.2 FAL-AR PDU Abstract Syntax 2

The productions defined here shall be used with the Traditional Encoding Rules to maintain compatibility with prior standards.

```
FalArPDU ::=          ConfirmedSend-RequestPDU
                      || ConfirmedSend-ResponsePDU
                      || UnconfirmedSend-PDU
                      || UnconfirmedAcknowledgedSend-CommandPDU
                      || UnconfirmedAcknowledgedSend-AffirmativePDU
                      II Idle-CommandPDU
                      II AR-XON-OFF-CommandPDU
                      || Establish-RequestPDU
                      || Establish-Request2PDU
                      || Establish-ResponsePDU
                      || Establish-ErrorPDU
                      || Abort-PDU
```

4.2.1 Confirmed Send Service

```
ConfirmedSend-RequestPDU ::= SEQUENCE {
      FalArHeader,
      confirmed-RequestPDU                     [1] IMPLICIT SEQUENCE {
           InvokeID,
           ConfirmedServiceRequest
      }
}

ConfirmedSend-ResponsePDU ::= SEQUENCE {
      FalArHeader,
      pduBody         CHOICE {
           confirmed-ResponsePDU               [2] IMPLICIT SEQUENCE {
                InvokeID,
                ConfirmedServiceResponse
           },
           confirmed-ErrorPDU                  [3] IMPLICIT SEQUENCE {
                InvokeID,
                ConfirmedServiceError
           }
      }
}
```

4.2.2 Unconfirmed Send Service

```
UnconfirmedSend-PDU ::= SEQUENCE {
      FalArHeader,
      unconfirmed-PDU                          [4] IMPLICIT SEQUENCE {
           InvokeID,
           UnconfirmedServiceRequest
      }
}
```

4.2.3 Unconfirmed Acknowledged Send Service

```
UnconfirmedAcknowledgedSend-CommandPDU ::= SEQUENCE {
      FalArHeader,
      unconfirmed-PDU                          [4] IMPLICIT SEQUENCE {
           InvokeID,
           UnconfirmedServiceRequest
}

UnconfirmedAcknowledgedSend-AffirmativePDU::= {
      FalArHeader
}
```

4.2.4 Idle Send Service

```
IdleSend-CommandPDU::= {
      FalArHeader
}
```

4.2.5 AR-XON-OFF Send Service

```
AR-XON-OFF-CommandPDU ::= SEQUENCE {
      FalArHeader,
      AR-XON-OFFPDU                            [7] IMPLICIT SEQUENCE {
           Invoke ID,
           XON-OFF
      }
}
```

4.2.6 Establish Service

```
Establish-RequestPDU ::= SEQUENCE {
      FalArHeader,
      RespondingAREP,
      RequestingAREP,
      userData                                   [6] IMPLICIT SEQUENCE {
           InvokeID,
           InitiateRequest                       [0]  IMPLICIT Initiate-RequestPDU
      }
}

Establish-Request2PDU ::= SEQUENCE {
      FalArHeader,
      RespondingAREP,
      RequestingAREP,
      AREPContext                                [5] IMPLICIT SEQUENCE {
           MaxOSCC,
           MaxOSCS,
           MAXUCSC,
           MAXUCSS,
           CIU
      }
}

Establish-ResponsePDU ::= SEQUENCE {
      FalArHeader,
      userData                                   [6] IMPLICIT SEQUENCE {
           InvokeID,
           InitiateResponse                      [1]  IMPLICIT Initiate-ResponsePDU
      }
}

Establish-ErrorPDU ::= SEQUENCE {
      FalArHeader,
      userData                                   [6] IMPLICIT SEQUENCE {
           InvokeID,
           initiateError                         [2]  IMPLICIT Initiate-ErrorPDU
      }
}
```

4.2.7 MaxOSCC

MaxOSCC ::= Unsigned8

4.2.8 MaxOSCS

MaxOSCS ::= Unsigned8

4.2.9 MaxUCSC

MaxUCC ::= Unsigned8

4.2.10 MaxUCSS

MaxUCS ::= Unsigned8

4.2.11 XON_OFF

XON_OFF ::= Unsigned8

4.2.12 CIU

CIU ::= Unsigned32

4.3 Abstract Syntax of PDUBody

4.3.1 Abort Service

```
Abort-PDU ::= SEQUENCE {
    FalArHeader,
    Identifier,
    ReasonCode,
    AdditionalDetail
}
```

4.3.2 InvokeID

```
InvokeID ::=  Unsigned8
```

4.3.3 ConfirmedServiceRequest

```
ConfirmedServiceRequest ::= CHOICE {
    status-Request                              [0] IMPLICIT Status-RequestPDU,
    identify-Request                            [1] IMPLICIT Identify-RequestPDU,
    read2-Request                               [2] IMPLICIT Read2-RequestPDU,
    write-Request                               [3] IMPLICIT Write-RequestPDU,
    getAttributes2-Request                      [4] IMPLICIT GetAttributes2-RequestPDU,
    readWithType-Request                        [5] IMPLICIT ReadWithType-RequestPDU,
    writeWithType-Request                       [6] IMPLICIT WriteWithType-RequestPDU,
    defineVariableList-Request                  [7] IMPLICIT DefineVariableList-RequestPDU,
    deleteVariableList-Request                  [8] IMPLICIT DeleteVariableList-RequestPDU,
    initiateClientPullDownloadStatic-Request    [9] IMPLICIT InitiateClientPullDownloadStatic-RequestPDU,
    pullDownloadSegment-Request                 [10] IMPLICIT PullDownloadSegment-RequestPDU,
    terminatePullDownload-Request               [11] IMPLICIT TerminatePullDownload-RequestPDU,
    initiateClientPullUploadStatic-Request      [12] IMPLICIT InitiateClientPullUploadStatic-Request,
    pullUploadSegment-Request                   [13] IMPLICIT PullUploadSegment-RequestPDU,
    terminatePullUpload-Request                 [14] IMPLICIT TerminatePullUpload-RequestPDU,
    initiateServerPullDownload-Request          [15] IMPLICIT InitiateServerPullDownload-RequestPDU,
    initiateServerPullUpload-Request            [16] IMPLICIT InitiateServerPullUpload-RequestPDU,
    createFunctionInvocation-Request            [17] IMPLICIT CreateFunctionInvocation-RequestPDU,
    deleteFunctionInvocation-Request            [18] IMPLICIT DeleteFunctionInvocation-RequestPDU,
    start-Request                               [19] IMPLICIT Start-RequestPDU,
    stop-Request                                [20] IMPLICIT Stop-RequestPDU,
    resume-Request                              [21] IMPLICIT Resume-RequestPDU,
    reset-Request                               [22] IMPLICIT Reset-RequestPDU,
    kill-Request                                [23] IMPLICIT Kill-RequestPDU,
    enableEvent-Request                         [24] IMPLICIT EnableEvent-RequestPDU,
    acknowledgeEventNotification-Request        [25] IMPLICIT AcknowledgeEventNotification-RequestPDU,
    physicalRead-Request                        [26] IMPLICIT PhysicalRead-RequestPDU,
    physicalWrite-Request                       [27] IMPLICIT PhysicalWrite-RequestPDU,
    beginSetAttributes-Request                  [28] IMPLICIT BeginSetAttributes-RequestPDU,
    setAttributes2-Request                      [29] IMPLICIT SetAttributes2-RequestPDU,
    endSetAttributes-Request                    [30] IMPLICIT EndSetAttributes-RequestPDU,
    initiateClientPushDownload-Request          [31] IMPLICIT InitiateClientPushDownload-RequestPDU,
    pushDownloadSegment-Request                 [32] IMPLICIT PushDownloadSegment-RequestPDU,
    terminatePushDownload-Request               [33] IMPLICIT TerminatePushDownload-RequestPDU,
    discard-Request                             [34] IMPLICIT Discard-RequestPDU,
    readList-Request                            [35] IMPLICIT ReadList-RequestPDU,
    writeList-Request                           [36] IMPLICIT WriteList-RequestPDU,
    exchange-Request                            [37] IMPLICIT Exchange-RequestPDU,
    getEventSummary-Request                     [38] IMPLICIT GetEventSummary-RequestPDU,
    getEventSummaryList-Request                 [39] IMPLICIT GetEventSummaryList-RequestPDU,
    setAttributes1-Request                      [40] IMPLICIT SetAttributes1-RequestPDU,
    getAttributes1-Request                      [41] IMPLICIT GetAttributes1-RequestPDU,
    read1-Request                               [42] IMPLICIT Read1-RequestPDU,
    conclude-Request                            [43]IMPLICIT Conclude-RequestPDU,
    subscribe-Request                           [44] IMPLICIT Subscribe-RequestPDU,
    initiateClientPullDownloadDynamic-Reques    [45] IMPLICIT InitiateClientPullDownloadDynamic-RequestPDU,
    initiateClientPullUpoadDynamic-Request      [46] IMPLICIT InitiateClientPullUploadDynamic-RequestPDU,
    exchangeList-Request                        [47] IMPLICIT ExchangeList-RequestPDU,
    enableEventList-Request                     [48] IMPLICIT EnableEventList-RequestPDU,
    confirmedAcknowledgeEventList-Request       [49] IMPLICIT ConfirmedAcknowledgeEventList-RequestPDU,
    getAttributes3-Request                      [50] IMPLICIT GetAttributes3-RequestPDU,
    queryEventSummaryList-Request               [51] IMPLICIT QueryEventSummaryList-RequestPDU
}
```

4.3.4 ConfirmedServiceResponse

```
ConfirmedServiceResponse ::= CHOICE {
    status-Response                          [0] IMPLICIT Status-ResponsePDU,
    identify-Response                        [1] IMPLICIT Identify-ResponsePDU,
    read2-Response                           [2] IMPLICIT Read2-ResponsePDU,
    write-Response                           [3] IMPLICIT Write-ResponsePDU,
    getAttributes2-Response                  [4] IMPLICIT GetAttributes2-ResponsePDU,
    readWithType-Response                    [5] IMPLICIT ReadWithType-ResponsePDU,
    writeWithType-Response                   [6] IMPLICIT WriteWithType-ResponsePDU,
    defineVariableList-Response              [7] IMPLICIT DefineVariableList-ResponsePDU,
    deleteVariableList-Response              [8] IMPLICIT DeleteVariableList-ResponsePDU,
    initiateClientPullDownloadStatic-Response [9] IMPLICIT InitiateClientPullDownloadStatic-ResponsePDU,
    pullDownloadSegment-Response             [10] IMPLICIT PullDownloadSegment-ResponsePDU,
    terminatePullDownload-Response           [11] IMPLICIT TerminatePullDownload-ResponsePDU,
    initiateClientPullUploadStatic-Response  [12] IMPLICIT InitiateClientPullUploadStatic-ResponsePDU,
    pullUploadSegment-Response               [13] IMPLICIT PullUploadSegment-ResponsePDU,
    terminatePullUpload-Response             [14] IMPLICIT TerminatePullUpload-ResponsePDU,
    initiateServerPullDownload-Response      [15] IMPLICIT InitiateServerPullDownload-ResponsePDU,
    initiateServerPullUpload-Response        [16] IMPLICIT InitiateServerPullUpload-ResponsePDU,
    createFunctionInvocation-Response        [17] IMPLICIT CreateFunctionInvocation-ResponsePDU,
    deleteFunctionInvocation-Response        [18] IMPLICIT DeleteFunctionInvocation-ResponsePDU,
    start-Response                           [19] IMPLICIT Start-ResponsePDU,
    stop-Response                            [20] IMPLICIT Stop-ResponsePDU,
    resume-Response                          [21] IMPLICIT Resume-ResponsePDU,
    reset-Response                           [22] IMPLICIT Reset-ResponsePDU,
    kill-Response                            [23] IMPLICIT Kill-ResponsePDU,
    enableEvent-Response                     [24] IMPLICIT EnableEvent-ResponsePDU,
    acknowledgeEventNotification-Response    [25] IMPLICIT AcknowledgeEventNotification-ResponsePDU,
    physicalRead-Response                    [26] IMPLICIT PhysicalRead-ResponsePDU,
    physicalWrite-Response                   [27] IMPLICIT PhysicalWrite-ResponsePDU,
    beginSetAttributes-Response              [28] IMPLICIT BeginSetAttributes-ResponsePDU,
    setAttributes2-Response                  [29] IMPLICIT SetAttributes2-ResponsePDU,
    endSetAttributes-Response                [30] IMPLICIT EndSetAttributes-ResponsePDU,
    initiateClientPushDownload-Response      [31] IMPLICIT InitiateClientPushDownload-ResponsePDU,
    pushDownloadSegment-Response             [32] IMPLICIT PushDownloadSegment-ResponsePDU,
    terminatePushDownload-Response           [33] IMPLICIT TerminatePushDownload-ResponsePDU,
    discard-Response                         [34] IMPLICIT Discard-ResponsePDU,
    readList-Response                        [35] IMPLICIT ReadList-ResponsePDU,
    writeList-Response                       [36] IMPLICIT WriteList-ResponsePDU,
    exchange-Response                        [37] IMPLICIT Exchange-ResponsePDU,
    getEventSummary-Response                 [38] IMPLICIT GetEventSummary-ResponsePDU,
    getEventSummaryList-Response             [39] IMPLICIT GetEventSummaryList-ResponsePDU,
    setAttributes1-Response                  [40] IMPLICIT SetAttributes1-ResponsePDU,
    getAttributes1-Response                  [41] IMPLICIT GetAttributes1-ResponsePDU,
    read1-Response                           [42] IMPLICIT Read1-ResponsePDU,
    conclude-Response                        [43]IMPLICIT Conclude-ResponsePDU,
    subscribe-Response                       [44] IMPLICIT Subscribe-ResponsePDU,
    initiateClientPullDownloadDynamic-Response [45] IMPLICIT InitiateClientPullDownloadDynamic-ResponsePDU,
    initiateClientPullUploadDynamic-Response [46] IMPLICIT InitiateClientPullUploadDynamic-ResponsePDU,
    exchangeList-Response                    [47] IMPLICIT ExchangeList-ResponsePDU,
    enableEventList-Response                 [48] IMPLICIT EnableEventList-ResponsePDU,
    confirmedAcknowledgeEventList-Response   [49] IMPLICIT ConfirmedAcknowledgeEventList-ResponsePDU,
    getAttributes3-Response                  [50] IMPLICIT GetAttributes3-ResponsePDU,
    queryEventSummaryList-Response           [51] IMPLICIT QueryEventSummaryList-ResponsePDU
}
```

4.3.5 ConfirmedServiceError

```
ConfirmedServiceError ::= CHOICE {
      status-Error                              [0] IMPLICIT ErrorType,
      identify-Error                            [1] IMPLICIT ErrorType,
      read2-Error                               [2] IMPLICIT ErrorType,
      write-Error                               [3] IMPLICIT ErrorType,
      getAttributes2-Error                      [4] IMPLICIT ErrorType,
      readWithType-Error                        [5] IMPLICIT ErrorType,
      writeWithType-Error                       [6] IMPLICIT ErrorType,
      defineVariableList-Error                  [7] IMPLICIT ErrorType,
      deleteVariableList-Error                  [8] IMPLICIT ErrorType,
      initiateClientPullDownloadStatic-Error    [9] IMPLICIT ErrorType,
      pullDownloadSegment-Error                 [10] IMPLICIT ErrorType,
      terminatePullDownload-Error               [11] IMPLICIT ErrorType,
      initiateClientPullUploadStatic-Error      [12] IMPLICIT ErrorType,
      pullUploadSegment-Error                   [13] IMPLICIT ErrorType,
      terminatePullUpload-Error                 [14] IMPLICIT ErrorType,
      initiateServerPullDownload-Error          [15] IMPLICIT ErrorType,
      initiateServerPullUpload-Error            [16] IMPLICIT ErrorType,
      createFunctionInvocation-Error            [17] IMPLICIT ErrorType,
      deleteFunctionInvocation-Error            [18] IMPLICIT ErrorType,
      start-Error                               [19] IMPLICIT ErrorType,
      stop-Error                                [20] IMPLICIT ErrorType,
      resume-Error                              [21] IMPLICIT ErrorType,
      reset-Error                               [22] IMPLICIT ErrorType,
      kill-Error                                [23] IMPLICIT ErrorType,
      enableEvent-Error                         [24] IMPLICIT ErrorType,
      acknowledgeEventNotification-Error        [25] IMPLICIT ErrorType,
      physicalRead-Error                        [26] IMPLICIT ErrorType,
      physicalWrite-Error                       [27] IMPLICIT ErrorType,
      beginSetAttributes-Error                  [28] IMPLICIT ErrorType,
      setAttributes2-Error                      [29] IMPLICIT ErrorType,
      endSetAttributes-Error                    [30] IMPLICIT ErrorType,
      initiateClientPushDownload-Error          [31] IMPLICIT ErrorType,
      pushDownloadSegment-Error                 [32] IMPLICIT ErrorType,
      terminatePushDownload-Error               [33] IMPLICIT ErrorType,
      discard-Error                             [34] IMPLICIT ErrorType,
      readList-Error                            [35] IMPLICIT ErrorType,
      writeList-Error                           [36] IMPLICIT ErrorType,
      exchange-Error                            [37] IMPLICIT ErrorType,
      getEventSummary-Error                     [38] IMPLICIT ErrorType,
      getEventSummaryList-Error                 [39] IMPLICIT ErrorType,
      setAttributes1-Error                      [40] IMPLICIT ErrorType,
      getAttributes1-Error                      [41] IMPLICIT ErrorType,
      read1-Error                               [42] IMPLICIT ErrorType,
      conclude-Error                            [43]IMPLICIT ErrorType,
      subscribe-Error                           [44] IMPLICIT ErrorType,
      initiateClientPullDownloadDynamic-Error   [45] IMPLICIT ErrorType,
      initiateClientPullDownloadDynamic-Error   [46] IMPLICIT ErrorType,
      exchangeList-Error                        [47] IMPLICIT ErrorType,
      enableEventList-Error                     [48] IMPLICIT ErrorType,
      confirmedAcknowledgeEventList-Error       [49] IMPLICIT ErrorType,
      getAttributes3-Error                      [50] IMPLICIT ErrorType,
      queryEventSummaryList-Error               [51] IMPLICIT ErrorType,
      xon-off-Error                             [52] IMPLICIT ErrorType
}
```

4.3.6 Error Type

```
ErrorType ::= SEQUENCE {
      errorClass                 [0] IMPLICIT ErrorClass,
      optionalParametersMap      [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      additionalCode             [1] IMPLICIT Integer16 OPTIONAL,
      additionalDescription      [2] IMPLICIT VisibleString OPTIONAL,
      serviceSpecificInfo        [3] IMPLICIT ANY OPTIONAL
}
```

4.3.7 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)
      }
}
```

4.3.8 Unconfirmed PDUs

```
UnconfirmedServiceRequest ::= CHOICE {
    informationReport-Request          [0] IMPLICIT InformationReport-RequestPDU,
    unsolicitedStatus-Request          [1] IMPLICIT UnsolicitedStatus-RequestPDU,
    eventNotification-Request          [2] IMPLICIT EventNotification-RequestPDU,
    informationReportWithType          [3] InformationReportWithType-RequestPDU,
    eventNotificationWithType          [4] EventNotificationWithType-RequestPDU,
    actionInvoke-Request               [5] IMPLICIT ActionInvoke-RequestPDU,
    actionReturn-Request               [6] IMPLICIT ActionReturn-RequestPDU,
    informationReportList-Request      [7] IMPLICIT InformationReportList-RequestPDU,
    reject-Request                     [8] IMPLICIT Reject-RequestPDU
    notificationRecovery-Request       [9] IMPLICIT NotificationRecovery-RequestPDU,
    eventNotificationList-Request      [10] IMPLICIT EventNotificationList-RequestPDU
 }
```

4.3.9 Management ASE

4.3.9.1 Begin Set Attributes Service

```
BeginSetAttributes-RequestPDU ::= Integer8 {
    loadingNonInteracting              (0),
    appendingInteracting               (1),
    freshLoadingInteracting            (2)
}
```

BeginSetAttributes -ResponsePDU ::= NULL

4.3.9.2 Create Service

4.3.9.2.1 Create Function Invocation PDU

```
CreateFunctionInvocation-RequestPDU ::= SEQUENCE {
    listOfRelatedObjects               [0] IMPLICIT SEQUENCE OF Gn_AccessSpecification,
    accessPrivilege                    [1] IMPLICIT Fi_AccessPrivilege,
    optionalParametersMap              [10] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
    fiName                             [2] IMPLICIT Gn_Name OPTIONAL,                -- bit 1
    fiIndex                            [12] IMPLICIT Gn_NumericID OPTIONAL,          -- bit 2
    userDescription                    [3] IMPLICIT ANY OPTIONAL,                     -- bit 3
    reusable                           [4] IMPLICIT Gn_Reusable DEFAULT TRUE,         -- bit 4
    deletable                          [5] IMPLICIT Gn_Deletable OPTIONAL,            -- bit 5
    startServiceArgumentType           [6] IMPLICIT Gn_NumericID OPTIONAL,            -- bit 6
    stopServiceArgumentType            [7] IMPLICIT Gn_NumericID OPTIONAL,            -- bit 7
    resumeServiceArgumentType          [8] IMPLICIT Gn_NumericID OPTIONAL,            -- bit 8
    resetServiceArgumentType           [9] IMPLICIT Gn_NumericID OPTIONAL,            -- bit 9
    killServiceArgumentType            [11] IMPLICIT Gn_NumericID OPTIONAL            -- bit 10
```

CreateFunctionInvocation-ResponsePDU ::= Gn_NumericID

4.3.9.2.2 Define Variable List PDU

```
DefineVariableList-RequestPDU ::= SEQUENCE {
    listOfVariables                    [0] IMPLICIT SEQUENCE OF Gn_AccessSpecification,
    accessPrivilege                    [1] IMPLICIT Vr_AccessPrivilege,
    optionalParametersMap              [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    variableListName                   [2] IMPLICIT Gn_Name OPTIONAL,                 -- bit 1
    variableListIndex                  [5] IMPLICIT Gn_NumericID OPTIONAL,            -- bit 2
    userDescription                    [3] IMPLICIT ANY OPTIONAL,                     -- bit 3
    deletable                          [4] IMPLICIT Gn_Deletable OPTIONAL             -- bit 4
}
```

DefineVariableList-ResponsePDU ::= Gn_NumericID

4.3.9.3 Delete Service

4.3.9.3.1 Delete Function Invocation PDU

DeleteFunctionInvocation-RequestPDU ::= Gn_AccessSpecification

DeleteFunctionInvocation-ResponsePDU ::= NULL

4.3.9.3.2 Delete Variable List PDU

DeleteVariableList-RequestPDU ::= Gn_AccessSpecification

DeleteVariableList-ResponsePDU ::= NULL

4.3.9.4 End Set Attributes Service

EndSetAttributes-RequestPDU ::= NULL

EndSetAttributes-ResponsePDU ::= NULL

4.3.9.5 Get Attributes List Service

```
GetAttributes1-RequestPDU ::= CHOICE {
      listOfObjectsAndAttributes                  [0] IMPLICIT SEQUENCE OF SEQUENCE {
           falClass                               [6] IMPLICIT Gn_NumericID,
           keyAttribute                                Gn_KeyAttribute,
           listOfRequestedAttributes              [7] IMPLICIT Mn_AttributesMap
      },
      rangeOfObjectsAndAttributes                 [1] IMPLICIT SEQUENCE {
           startingNumericId                      [0] IMPLICIT Gn_NumericID,
           listOfRequestedAttributes              [1] IMPLICIT Mn_AttributesMap
      }
}

GetAttributes2-RequestPDU ::= SEQUENCE {
      listOfAttributes                            [0] IMPLICIT Mn_SelectedAttributes,
      accessSpecification  CHOICE {
           numericID                              [1] IMPLICIT Gn_NumericID,
           variableName                           [2] IMPLICIT Gn_Name,
           variableListName                       [3] IMPLICIT Gn_Name,
           domainName                             [4] IMPLICIT Gn_Name,
           fiName                                 [5] IMPLICIT Gn_Name,
           eventName                              [6] IMPLICIT Gn_Name,
           startIndex                             [7] IMPLICIT Gn_NumericID
      }
}

GetAttributes3-RequestPDU ::= SEQUENCE {
      objectClass                                 [2] IMPLICIT Gn_ObjectClass,
      start objectID                                 Gn_AccessSpecification
}

GetAttributes1-ResponsePDU  ::=  SEQUENCE  {
      listOfAttributeIdAndValues                  [0] IMPLICIT Mn_AttributeValues,
      more                                        [1] IMPLICIT Gn_MoreFollows
}

GetAttributes2-ResponsePDU ::= SEQUENCE {
      listOfObjectDefinition                      [0] IMPLICIT SEQUENCE OF Gn_ObjectDefinition,
      more                                        [1] IMPLICIT Gn_MoreFollows
  }

GetAttributes3-ResponsePDU ::= CHOICE {
      listofobjectID                              [0] IMPLICIT SEQUENCE {
           listofID                               [2] IMPLICIT SEQUENCE OF Gn_AccessSpecification,
           morefollows                            [3] IMPLICIT Gn_MoreFollows
      } ,  --The object class value in the request is <> 0
      listofobjectDefinition                      [1] IMPLICIT SEQUENCE {
           listofDefinition                       [4] IMPLICIT SEQUENCE OF Mn_AttributesValues
           morefollows                            [5] IMPLICIT Gn-MoreFollows
      }    --The object classvalue in the request is = 0
}
```

4.3.9.6 Set Attributes Service

```
SetAttributes1-RequestPDU  ::=  SEQUENCE OF SEQUENCE {
      falClass                                       [6] IMPLICIT Gn_NumericID,
      keyAttributes                                  Gn_KeyAttributes,
      listOfAttributeUpdates                      [7] IMPLICIT Mn_AttributeValues
}

SetAttributes2-RequestPDU ::= SEQUENCE OF {
      objectDefinition                               [0] IMPLICIT Gn_ObjectDefinition
}
```

```
SetAttributes1-ResponsePDU  ::=  NULL

SetAttributes2-ResponsePDU  ::=  NULL
```

4.3.10 Application Process ASE

4.3.10.1 Get Status Service

```
Status-RequestPDU ::= NULL

Status-ResponsePDU ::= SEQUENCE {
    logicalStatus                       [0] IMPLICIT Unsigned8 {
        state-changes-allowed                (0),
        no-state-changes-allowed             (1),
        limited-services-permitted           (2),
        od-loading-non-interacting           (4),
        od-loading-interacting               (5)
    },
    physicalStatus                      [1] IMPLICIT Unsigned8 {
        operational                          (0),
        partially-operational                (1),
        inoperable                           (2),
        needs-commissioning                  (3)
    },
    optionalParametersMap               [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    localDetail                         [2] IMPLICIT BitString OPTIONAL                      --bit 1
 }
```

4.3.10.2 Identify Service

```
Identify-RequestPDU ::= NULL

Identify-ResponsePDU ::= SEQUENCE {
    vendorName                          [0]  IMPLICIT VisibleString,
    modelIdentifier                     [1]  IMPLICIT VisibleString,
    vendorRevision                      [2]  IMPLICIT VisibleString,
    optionalParametersMap               [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    serialNumber                        [3]  IMPLICIT OctetString OPTIONAL                   -- bit 1
}
```

4.3.10.3 Initiate Service

```
Initiate-RequestPDU ::= SEQUENCE {
    versionObjectDefinitionsCalling           [0] IMPLICIT Integer16,
    apDescriptorCalling                       [1] IMPLICIT OctetString,
    accessProtectionSupportedCalling          [2] IMPLICIT Ap_AccessProtectionSupported,
    passwordAndAccessGroupsCalling            [3] IMPLICIT Ap_AccessControl,
    configuredMaxPduSizeSending               [4] IMPLICIT Unsigned8,              -- See NOTE
    configuredMaxPduSizeReceiving             [5] IMPLICIT Unsigned8,              -- See NOTE
    listOfSupportedServicesCalling            [6] IMPLICIT Mn_PduSupportedMap,
    configuredMaxOutstandingServicesRequesting [7] IMPLICIT Unsigned8,
    configuredMaxOutstandingServicesResponding [8] IMPLICIT Unsigned8,
    optionalParametersMap                     [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    userDetail                                [11] IMPLICIT OctetString OPTIONAL,    -- bit 1
    configuredNestingLevel                    [12] IMPLICIT Unsigned8  DEFAULT 1      -- bit 2
}

Initiate-ResponsePDU ::= SEQUENCE {
    versionObjectDefinitionsCalled            [0] IMPLICIT Integer16,
    profileNumberCalled                       [1] IMPLICIT OctetString,
    accessPrivilegeSupportedCalled            [2] IMPLICIT Ap_AccessProtectionSupported,
    passwordAndAccessGroupsCalled             [3] IMPLICIT Ap_AccessControl,
    negotiatedMaxOutstandingServicesRequesting [4] IMPLICIT Unsigned8,
    negotiatedMaxOutstandingServicesResponding [5] IMPLICIT Unsigned8
    optionalParametersMap                     [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    userDetail                                [11] IMPLICIT OctetString OPTIONAL,    -- bit 1
    negotiatedNestingLevel                    [12] IMPLICIT Unsigned8 DEFAULT 1,     -- bit 2
    negotiatedListOfServices                  [13] IMPLICIT Mn_PduSupportedMap OPTIONAL          -- bit 3
 }
```

```
Initiate-ErrorPDU ::= SEQUENCE {
      errorCode                          [0] IMPLICIT  Integer8 {
           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          [1] IMPLICIT Unsigned8,            -- See NOTE
      maxPduLengthReceivingCalled        [2] IMPLICIT Unsigned8,            -- See NOTE
      listOfSupportedServicesCalled      [3] IMPLICIT Mn_PduSupportedMap
 }
-- NOTE   For QUB-Seg the PDU length is given in multiples of 256 octets, i.e. value 4 means 4*256 = 1024 octets
```

4.3.10.4 Status Notification Service

UnsolicitedStatus-RequestPDU ::= Status-ResponsePDU

4.3.10.5 Subscribe Service

```
Subscribe-RequestPDU  ::=  CHOICE  {
      joining                            [0] IMPLICIT SEQUENCE  {
           publishedObject                          Gn_KeyAttributes,
           scheduleInterval                   [11] IMPLICIT Dl_CyclicInterval,
           optionalParametersMap              [12] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
           maximumARs                         [13] IMPLICIT Unsigned8 DEFAULT 1 OPTIONAL,        -- bit 1
           maximumPermissibleJitter           [14] IMPLICIT Dl_PermissibleJitter OPTIONAL,        -- bit 2
           desiredPublishingStartTime         [15] IMPLICIT FieldbusTime OPTIONAL,                -- bit 3
           earliestPublishingStartTime        [16] IMPLICIT FieldbusTime OPTIONAL,                -- bit 4
           latestPublishingStartTime          [17] IMPLICIT FieldbusTime OPTIONAL                 -- bit 5
      },
      listOfLeavingAREPs                 [1] IMPLICIT SEQUENCE OF Gn_NumericID
}

Subscribe-ResponsePDU  ::=  SEQUENCE  {
      optionalParametersMap              [0] Gn_OptionalParametersMap8 OPTIONAL,
      dataType                           [1] Gn_NumericID OPTIONAL,                               -- bit 1
      dataLength                         [2] Dl_SduSize  OPTIONAL,                                -- bit 2
      dedicated                          [3] Ar_Dedicated  DEFAULT TRUE,                          -- bit 3
      listOfArInformation                     [4] IMPLICIT SEQUENCE OF SEQUENCE  {
           publishingAREPid                   [0] Gn_NumericID,
           publishingDlMapping                [1] Gn_NumericID,
           scheduledStartTime                 [2] FieldbusTime
      }  OPTIONAL                                                                                 -- bit 4
}
```

4.3.10.6 Reject Service

```
RejectPDU  ::=       SEQUENCE  {
      original-invokeID                  [0] IMPLICIT Integer8,
      reject-code                        [1] IMPLICIT Integer8 {
           other                              (0),
           illegal-confirmed-service-request  (1),
           pdu-error                          (2),
           pdu-size                           (5)
      }
}
```

4.3.10.7 Conclude Service

Conclude-Request ::= Null

Conclude-Response ::= Null

4.3.11 Load Region ASE

4.3.11.1 Discard Service

```
Discard-RequestPDU  ::=  Gn_KeyAttribute                  --  Load Region LocalID
```

Discard-ResponsePDU ::= NULL

4.3.11.2 Initiate Load Service

4.3.11.2.1 InitiateClientPullDownloadStatic PDUs

InitiateClientPullDownloadStatic-RequestPDU ::= Gn_AccessSpecification

InitiateClientPullDownloadStatic-ResponsePDU ::= NULL

4.3.11.2.2 InitiateClientPullDownloadDynamic PDUs

```
InitiateClientPullDownloadDynamic-RequestPDU ::= SEQUENCE {
      loadRegionKeyAttribute                      Gn_AccessSpecification,
      userData                                    [0]  OctetString,
      sharable                                    [1]  BOOLEAN,
      accessProtection                            [2]  Fi-AccessPrivilege
}
```

InitiateClientPullDownloadDynamic-ResponsePDU ::= NULL

4.3.11.2.3 InitiateClientPullUploadStatic PDUs

InitiateClientPullUploadStatic-RequestPDU ::= Gn_AccessSpecification

InitiateClientPullUploadStatic-ResponsePDU ::= NULL

4.3.11.2.4 InitiateClientPullUploadDynamic PDUs

InitiateClientPullUploadDynamic-RequestPDU ::= Gn_AccessSpecification

```
InitiateClientPullUploadDynamic-ResponsePDU ::= SEQUENCE {
      ulsmNumericID                               [0]  Gn_NumericID,
      userData                                    [1]  OctetString
}
```

4.3.11.2.5 InitiateServerPullDownload PDUs

```
InitiateServerPullDownload-RequestPDU ::= SEQUENCE {
      loadRegionKeyAttribute                      Gn_AccessSpecification,
      optionalParametersMap                  [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      additionalInformation                  [11] IMPLICIT VisibleString OPTIONAL                  -- bit 1
}
```

InitiateServerPullDownload-ResponsePDU ::= NULL

4.3.11.2.6 InitiateServerPullUpload PDUs

```
InitiateServerPullUpload-RequestPDU ::= SEQUENCE {
      loadRegionKeyAttribute                      Gn_AccessSpecification,
      optionalParametersMap                  [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      additionalInformation                  [2] IMPLICIT VisibleString OPTIONAL                   -- bit 1
}
```

InitiateServerPullUpload-ResponsePDU ::= NULL

4.3.11.2.7 InitiateClientPushDownload PDUs

InitiateClientPushDownload-RequestPDU ::= Gn_AccessSpecification

InitiateClientPushDownload-ResponsePDU ::= NULL

4.3.11.3 Pull Segment Service

4.3.11.3.1 PullDownloadSegment PDUs

```
PullDownloadSegment-RequestPDU ::= SEQUENCE {
      loadRegionKeyAttribute                      Gn_AccessSpecification
      optionalParametersMap                  [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      segmentNumber                          [11] IMPLICIT Fi_SegmentNumber OPTIONAL                -- bit 1
}
```

```
PullDownloadSegment-ResponsePDU ::= SEQUENCE {
      loadData                               [0] IMPLICIT OctetString,
      moreFollows                            [1] IMPLICIT Gn_MoreFollows
}
```

4.3.11.3.2 PullUploadSegment PDUs

```
PullUploadSegment-RequestPDU ::= Gn_AccessSpecification

PullUploadSegment-ResponsePDU ::= SEQUENCE {
      loadData                                  [0] IMPLICIT OctetString,
      moreFollows                               [1] IMPLICIT Gn_MoreFollows
}
```

4.3.11.4 Push Segment Service

4.3.11.4.1 PushDownloadSegment PDUs

```
PushDownloadSegment-RequestPDU ::= SEQUENCE {
      loadRegionKeyAttribute                    Gn_AccessSpecification,
      loadData                                  [2] IMPLICIT OctetString,
      moreFollows                               [3] IMPLICIT Gn_MoreFollows,
      optionalParametersMap                     [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      segmentNumber                             [11] IMPLICIT Fi_SegmentNumber OPTIONAL          -- bit 1
}

PushDownloadSegment-ResponsePDU ::= NULL
```

4.3.11.5 Terminate Load Service

4.3.11.5.1 TerminatePullDownload PDUs

```
TerminatePullDownload-RequestPDU ::= SEQUENCE {
      loadRegionKeyAttribute                    Gn_AccessSpecification,
      terminateReason                           [2] IMPLICIT Lr_TerminateReason
}

TerminatePullDownload-ResponsePDU ::= NULL
```

4.3.11.5.2 TerminatePushDownload PDUs

```
TerminatePushDownload-RequestPDU ::= Gn_AccessSpecification

TerminatePushDownload-ResponsePDU ::= Fi_TerminateReason
```

4.3.11.5.3 TerminatePullUpload PDUs

```
TerminatePullUpload-RequestPDU ::= Gn_AccessSpecification

TerminatePullUpload-ResponsePDU ::= NULL
```

4.3.12 Function Invocation ASE

4.3.12.1 ActionInvoke Service

```
ActionInvoke-RequestPDU  ::=  SEQUENCE  {
      keyAttribute                              Gn_KeyAttribute,
      actionInvocationID                        [6] IMPLICIT Fi_ActionInvokeID,
      optionalParametersMap                     [10] IMPLICIT Gn_OptionalParametersMap8  OPTIONAL,
      executionArgument                         [7] IMPLICIT OctetString  OPTIONAL               -- bit 1
}
```

4.3.12.2 ActionReturn Service

```
ActionReturn-RequestPDU  ::=  SEQUENCE  {
      actionInvocationID                        [0] IMPLICIT Fi_ActionInvokeID,
      actionResults  CHOICE {
           actionSuccess                        [1] IMPLICIT ANY,
           actionFailure                        [2] IMPLICIT ErrorType
      }
}
```

4.3.12.3 Kill Service

```
Kill-RequestPDU ::= SEQUENCE {
      keyAttribute                                 Gn_KeyAttribute,
      optionalParametersMap                     [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      executionArgument                         [7] IMPLICIT OctetString OPTIONAL                -- bit 1
}

Kill-ResponsePDU ::= NULL
```

4.3.12.4 Reset Service

```
Reset-RequestPDU ::= SEQUENCE {
      keyAttribute                          Gn_KeyAttribute,
      optionalParametersMap           [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      executionArgument               [7] IMPLICIT OctetString OPTIONAL                  -- bit 1
}
```

```
Reset-ResponsePDU ::= NULL
```

4.3.12.5 Resume Service

```
Resume-RequestPDU ::= SEQUENCE {
      keyAttribute                          Gn_KeyAttribute,
      optionalParametersMap           [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      executionArgument               [7] IMPLICIT OctetString OPTIONAL                  -- bit 1
}
```

```
Resume-ResponsePDU ::= NULL
```

4.3.12.6 Start Service

```
Start-RequestPDU ::= SEQUENCE {
      keyAttribute                          Gn_KeyAttribute,
      optionalParametersMap           [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      executionArgument               [7] IMPLICIT OctetString OPTIONAL                  -- bit 1
}
```

```
Start-ResponsePDU ::= NULL
```

4.3.12.7 Stop Service

```
Stop-RequestPDU ::= SEQUENCE {
      keyAttribute                          Gn_KeyAttribute,
      optionalParametersMap           [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      executionArgument               [7] IMPLICIT OctetString OPTIONAL                  -- bit 1
}
```

```
Stop-ResponsePDU ::= NULL
```

4.3.13 Variable Access ASE

4.3.13.1 Exchange Service

```
Exchange-RequestPDU  ::=  SEQUENCE  {
      specifierForVariableToWrite     [0] Gn_KeyAttribute,
      valueToWrite                    [1] IMPLICIT ANY,
      specifierForVariableToRead      [2] Gn_KeyAttribute,
      optionalParametersMap           [2] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      objectRevisionForVariableWriting [3] IMPLICIT Gn_ObjectRevision  OPTIONAL,      -- bit 1
      objectRevisionRequested         [4] IMPLICIT Gn_RequestFlag DEFAULT FALSE       -- bit 2
}
```

```
Exchange-ResponsePDU  ::=  SEQUENCE  {
      writeResult                     [0] IMPLICIT Vr_WriteResult,
      readResult    CHOICE  {
            status                    [1] IMPLICIT Er_Access,
            value                     [2] IMPLICIT ANY,
            valueWithRevision         [3] IMPLICIT SEQUENCE {
                  value                     [4] IMPLICIT ANY,
                  objectRevision            [5] IMPLICIT Gn_ObjectRevision
            }
      }
}
```

4.3.13.2 Exchange List Service

```
ExchangeList-RequestPDU ::= SEQUENCE {
      writeList-Request                     [0] IMPLICIT WriteList-RequestPDU,
      readList-Request                      [1] IMPLICIT ReadList-RequestPDU
}
```

```
ExchangeList-Response ::= SEQUENCE {
      writeList-Response                 [0] IMPLICIT WriteList-ResponsePDU,
      readList-Response                  [1] IMPLICIT ReadList-ResponsePDU
}
```

4.3.13.3 Information Report Service

```
InformationReport-RequestPDU ::= SEQUENCE {
      value                              [4] IMPLICIT ANY,
      optionalParametersMap              [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      variableSpecifier                        Gn_KeyAttribute OPTIONAL,                   -- bit 1
      subIndex                           [3] IMPLICIT Gn_SubIndex OPTIONAL,               -- bit 2
      objectRevision                     [11] IMPLICIT Gn_ObjectRevision OPTIONAL         -- bit 3
}
```

4.3.13.4 Information Report with Type Service

```
InformationReportWithType-RequestPDU ::= SEQUENCE{
      TypeDescription                    [4] Gn_TypeDescription,
      value                              [5] IMPLICIT ANY,
      optionalParametersMap              [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      variableSpecifier                        Gn_KeyAttribute OPTIONAL,                   -- bit 1
      subIndex                           [3] IMPLICIT Gn_SubIndex OPTIONAL,               -- bit 2
      objectRevision                     [11] IMPLICIT Gn_ObjectRevision OPTIONAL         -- bit 3
 }
```

4.3.13.5 Information List Report Service

```
InformationReportList-RequestPDU ::= SEQUENCE {
      ListOfVariableSpecifier CHOICE {
         variableList                    [0] IMPLICIT Gn_KeyAttribute,
         listOfVariable                  [1] IMPLICIT Gn_ListOfVariableAccess
      },
      listOfDataType                     [7] IMPLICIT SEQUENCE OF Gn_FullyNestedTypeDescription,
      listOfData                         [8] IMPLICIT SEQUENCE OF ANY,
      optionalParametersMap              [10] IMPLICIT Gn_OptionalParametersMap8,
      userDetail                         [11] IMPLICIT OctetString OPTIONAL,              -- bit 1
      listOfObjectRevision               [12] SEQUENCE OF Gn_ObjectRevision OPTIONAL      -- bit 2
}
```

-- The listOfDataType should either contain one element per variable or should be empty. The listOfObjectRevision should contain one element per variable or should be absent.

4.3.13.6 Read Service

4.3.13.6.1 Read PDUs

```
Read1-RequestPDU ::= SEQUENCE {
      variableSpecifier                        Gn_KeyAttribute,
      optionalParametersMap              [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      objectRevisionRequested            [11] IMPLICIT Gn_RequestFlag DEFAULT FALSE OPTIONAL      -- bit 1
}

Read2-RequestPDU ::= SEQUENCE {
      variableSpecifier                        Gn_KeyAttribute,
      optionalParametersMap              [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      subIndex                           [3] IMPLICIT Gn_SubIndex OPTIONAL                -- bit 1
}

Read1-ResponsePDU ::= SEQUENCE {
      value                              [0] IMPLICIT ANY,
      optionalParametersMap              [1] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      objectRevision                     [2] IMPLICIT Gn_ObjectRevision OPTIONAL          -- bit 1
}

Read2-ResponsePDU ::= ANY
```

4.3.13.6.2 PhysicalRead PDUs

```
PhysicalRead-RequestPDU ::= SEQUENCE {
 local-address                          [0] IMPLICIT Gn_PhysicalAddress,
 length                                 [1] IMPLICIT Gn_Length
}
```

```
PhysicalRead-ResponsePDU ::= ANY
```

4.3.13.6.3 ReadWithType PDUs

```
ReadWithType-RequestPDU ::= SEQUENCE{
    variableSpecifier                          Gn_KeyAttribute,
    optionalParametersMap               [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    subIndex                            [3] IMPLICIT Gn_SubIndex OPTIONAL                  -- bit 1
 }
```

```
ReadWithType-ResponsePDU ::= SEQUENCE{
    typeDescription                     [0] Gn_TypeDescription,
    value                               [1] IMPLICIT ANY,
    optionalParametersMap               [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    objectRevision                      [11] IMPLICIT Gn_ObjectRevision OPTIONAL            -- bit 1
 }
```

4.3.13.7 Read List Service

```
ReadList-RequestPDU  ::=  SEQUENCE  {
    ListOfVariableSpecifier  CHOICE  {
        variableList                    [0] IMPLICIT Gn_KeyAttributes,
        listOfVariable                  [1] IMPLICIT Gn_ListOfVariableAccess,
        listOfNumericAddress            [2] IMPLICIT SEQUENCE OF Gn_NumericAddress
    },
    additionalInfoRequested             BitString8 {
        typeDescriptionRequested            (0),
        objectRevisionRequested             (1)
    }
}
```

```
ReadList-ResponsePDU  ::=  SEQUENCE  {
    ListOfAccessResult                  [0] IMPLICIT BitString,
    listOfDataType                      [1] IMPLICIT SEQUENCE OF Gn_FullyNestedTypeDescription,
    listOfData                          [2] IMPLICIT SEQUENCE OF ANY,
    listOfObjectRevision                [3] IMPLICIT SEQUENCE OF Gn_ObjectRevision
}
```

-- The listOfAccessResult parameter specifies for each requested variable if the access was successful (bit = 1) or not (bit = 0).

-- The listOfDataType should either contain one element per variable or should be empty. The same applies for the listOfObjectRevision.

-- The listOfData parameter specifies for each requested variable the value of the variable if the access succeeds or an ErrorClass if it fails. The ErrorClass is coded as an unsigned16 where the most significant byte contains the “error class” and the least significant byte contains the “error code” (see ErrorClass definition).

4.3.13.7.1 Write Service

4.3.13.7.2 Write PDUs

```
Write-RequestPDU  ::=  SEQUENCE  {
    variableSpecifier                          Gn_KeyAttribute,
    value                               [4] IMPLICIT ANY,
    optionalParametersMap               [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    subIndex                            [3] IMPLICIT Gn_SubIndex OPTIONAL,                 -- bit 1
    objectRevision                      [11] IMPLICIT Gn_ObjectRevision OPTIONAL            -- bit 2
}
```

```
Write-ResponsePDU  ::=  NULL
```

4.3.13.7.3 PhysicalWrite PDUs

```
PhysicalWrite-RequestPDU ::= SEQUENCE {
      local-address                     [0] IMPLICIT Gn_PhysicalAddress,
      data                              [1] IMPLICIT ANY
}

PhysicalWrite-ResponsePDU ::= NULL
```

4.3.13.7.4 WriteWithType PDUs

```
WriteWithType-RequestPDU ::= SEQUENCE{
      VariableSpecifier                      Gn_KeyAttribute,
      typeDescription                   [4] Gn_TypeDescription,
      value                             [5] IMPLICIT ANY,
      optionalParametersMap             [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      subIndex                          [3] IMPLICIT Gn_SubIndex OPTIONAL,              - bit 1
      objectRevision                    [11] IMPLICIT Gn_ObjectRevision OPTIONAL         - bit 2
 }

WriteWithType-ResponsePDU ::= NULL
```

4.3.13.8 Write List Service

```
WriteList-RequestPDU  ::=  SEQUENCE {
      ListOfVariableSpecifier  CHOICE  {
           variableList                 [0] IMPLICIT Gn_KeyAttributes,
           listOfVariable               [1] IMPLICIT Gn_ListOfVariableAccess,
           listOfNumericAddress         [2] IMPLICIT SEQUENCE OF Gn_NumericAddress
      },
      listOfDataType                    [3] IMPLICIT SEQUENCE OF Gn_FullyNestedTypeDescription,
      listOfValue                       [4] IMPLICIT SEQUENCE OF ANY,
      listOfObjectRevision              [5] IMPLICIT SEQUENCE OF Gn_ObjectRevision
}
```

-- The listOfDataType should either contain one element per variable or should be empty. The same applies for the listOfObjectRevision.

```
WriteList-ResponsePDU  ::=  SEQUENCE  {
      ListOfAccessResult                [0] IMPLICIT BitString,
      listOfAccessError                 [1] IMPLICIT SEQUENCE OF ErrorClass
}
```

-- The listOfAccessResult parameter specifies for each requested variable if the access was successful (bit = 1) or not (bit = 0).

-- The listOfAccessError contains one error code for each failed access. If a variable access was successful, there is no corresponding error code in this list.

-- The ErrorClass is coded as an unsigned16 where the most significant byte contains the “error class” and the least significant byte contains the “error code” (see ErrorClass definition).

4.3.14 Event Management ASE

4.3.14.1 Confirmed Acknowledge Event service

```
AcknowledgeEventNotification-RequestPDU  ::=  SEQUENCE {
      keyAttribute                          Gn_AccessSpecification,
      reportedEventCount                [2] IMPLICIT Ev_EventCount
}

AcknowledgeEventNotification-ResponsePDU ::= NULL
```

4.3.14.2 EnableEvent Service

```
EnableEvent-RequestPDU ::= SEQUENCE {
   keyAttribute                              Gn_AccessSpecification,
   enabled                              [2] IMPLICIT Gn_Enabled
}

EnableEvent-ResponsePDU ::= NULL
```

4.3.14.3 EnableEventList Service

```
EnableEventList-RequestPDU ::= SEQUENCE {
    listOfKeyAttribute                [0] IMPLICIT SEQUENCE OF Gn_AccessSpecification,
    listOfEnabledFlags                [1] IMPLICIT BitString
}

EnableEventList-ResponsePDU ::= SEQUENCE {
    listOfEnableResult                [0] IMPLICIT BitString,
    listOfError                       [1] IMPLICIT ErrorClass
}
```

4.3.14.4 Event Notification Service

```
EventNotificationList-RequestPDU  ::=  SEQUENCE {
    eventNotifierID                   IMPLICIT Gn_NumericID,
    eventDataContained                Boolean,
    notificationTypeInfo  CHOICE {
        basic                         [0] NULL,
        sequenced                     [1] Ev_SequenceNumber,
        time-of-notification          [2] Ev_TimeTag,
        compound                      [3] IMPLICIT SEQUENCE {
            sequence-number               Ev_SequenceNumber,
            time-tag                          Ev_TimeTag
        }
    },
    listOfEventID                     SEQUENCE OF Gn_AccessSpecification,
    listOfEv_EventDataType            SEQUENCE OF Gn_FullyNestedTypeDescription,
    listOfEventContents      CHOICE {
        simple                        [0] SEQUENCE OF ANY,    -- Null, Ev_EventData,
        reportedEventCount            [1] SEQUENCE OF ANY,    -- Ev_EventCount, Ev_EventData
        eventDetectionTime            [2] SEQUENCE OF ANY,    -- Ev_TimeTag, Ev_EventData
        composite                     [3] SEQUENCE OF ANY     -- Ev_EventCount, Ev_TimeTag, Ev_EventData
    }
}
```

```
--      If the eventDataContained parameter is TRUE, the Ev_EventData value shall be present.

--      The components of event contents shall be appended together without any gap.
```

```
EventNotification-RequestPDU ::= SEQUENCE {
    KeyAttribute                          Gn_KeyAttribute,
    eventNumber                       [2] IMPLICIT Unsigned8,
    optionalParametersMap             [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    eventData                         [3] IMPLICIT ANY OPTIONAL                                  -- bit 1
    eventNotifierID                   [11] IMPLICIT Gn_NumericID OPTIONAL,                       -- bit 2
    sequenceNumber                    [12] IMPLICIT Ev_SequenceNumber OPTIONAL,                  -- bit 3
    notificationTime                  [13] IMPLICIT Ev_TimeTag OPTIONAL                          -- bit 4
}
```

4.3.14.5 Event Notification with Type Service

```
EventNotificationWithType-RequestPDU ::= SEQUENCE{
    KeyAttribute                          Gn_KeyAttribute,
    eventNumber                       [2] IMPLICIT Unsigned8,
    typeDescription                   [3] Gn_TypeDescription,
    optionalParametersMap             [10] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    value                             [4] IMPLICIT ANY OPTIONAL,                                 -- bit 1
    eventNotifierID                   [11] IMPLICIT Gn_NumericID OPTIONAL,                       -- bit 2
    sequenceNumber                    [12] IMPLICIT Ev_SequenceNumber OPTIONAL,                  -- bit 3
    notificationTime                  [13] IMPLICIT Ev_TimeTag OPTIONAL                          -- bit 4
 }
```

4.3.14.6 Get Event Summary Service

```
GetEventSummary-RequestPDU  ::=  SEQUENCE  {
    eventID                           [0] IMPLICIT Gn_NumericID,
    optionalParametersMap             [1] IMPLICIT OptionalParametersMap8 OPTIONAL,
    recoveredMessageRequested         [2] IMPLICIT Gn_RequestFlag  OPTIONAL,                     -- bit 1
    objectRevisionRequested           [3] IMPLICIT Gn_RequestFlag  OPTIONAL                      -- bit 2
}
```

```
GetEventSummary-ResponsePDU  ::=  SEQUENCE  {
      eventStatus                           [0] IMPLICIT Ev_EventStatus,
      optionalParametersMap                 [1] IMPLICIT OptionalParametersMap8 OPTIONAL,
      objectRevision                        [2] IMPLICIT Gn_ObjectRevision  OPTIONAL,        -- bit 1
      recoveredMessage                      [3] IMPLICIT Ev_EventMessage OPTIONAL            -- bit 2
}
```

4.3.14.7 Get Event Summary List Service

```
GetEventSummaryList-RequestPDU ::= SEQUENCE {
      request events  CHOICE {
           allEvents                        [1] IMPLICIT Null,
           listofEvents                     [2] IMPLICIT SEQUENCE OF Gn_AccessSpecification
      } ,
      additionalInfoRequest                 [3] IMPLICIT BitString8 {
           recoveredMessageRequested             (0),  --true for requested
           objectRevisionRequested               (1)  --true for requested
      }
}

GetEventSummaryList-ResponsePDU ::= CHOICE {
      listofEvents                          [1] IMPLICIT SEQUENCE {
           listofAccessResult                    [0] IMPLICIT BitString,
           listofEventMessage                         Ev_ListOfEventResponse,
           listofObjectRevision                  [5]  IMPLICIT SEQUENCE OF Gn_ObjectRevision
      },
      allEvents                                  [2] IMPLICIT SEQUENCE {
           listofEventID                              [0] IMPLICIT SEQUENCE OF Gn_AccessSpecification,
           listofEventMessage                         Ev_ListOfEventResponse,
           listofObjectRevision                  [5]  IMPLICIT SEQUENCE OF Gn_ObjectRevision
      }
}
```

-- The listofAccessResult parameter specifies for each requested event, if the access was successful (bit=1) or not (bit=0).

4.3.14.8 Notification Recovery Service

```
NotificationRecovery-RequestPDU  ::=  SEQUENCE {
      notifierID                                 [0] IMPLICIT Gn_NumericID,
      optionalParametersMap                      [1] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      numberToRecover                            [2] IMPLICIT Ev_NumberToRecover  DEFAULT 1 OPTIONAL,      -- bit 1
      sequenceNumber                             [3] IMPLICIT Ev_SequenceNumber OPTIONAL                   -- bit 2
}
```

4.3.14.9 Confirmed Acknowledge Event List Service

```
ConfirmedAcknowledgedEventList-RequestPDU ::= SEQUENCE {
      acknowledgmentPolicy                  [0] IMPLICIT Ev_AcknowledgmentPolicy,
      listOfEventID                         [1] IMPLICIT SEQUENCE OF Gn_AccessSpecification,
      listOfMessageSpecifier  CHOICE {
           reportedEventCount               [2] IMPLICIT SEQUENCE OF Ev_EventCount,
           eventDetectionTime               [3] IMPLICIT SEQUENCE OF Ev_TimeTag,
           eventCountAndDetectionTime       [4] IMPLICIT SEQUENCE OF ANY  -- Ev_EventCount, Ev_TimeTag
      },
      listOfAcknowledgmentData              [5] IMPLICIT SEQUENCE OF OctetString
}

ConfirmedAcknowledgedEventList-ResponsePDU ::= SEQUENCE {
      listOfAccessResult                    [0] IMPLICIT BitString,
      listOfError                           [1] IMPLICIT ErrorClass
}
```

4.3.14.10 Query Event Summary List Service

```
QueryEventSummaryList-RequestPDU ::= SEQUENCE {
    eventselectionCriterion           [0] IMPLICIT BitString8,
    requestedEvent                    CHOICE {
        allEvents                     [1] IMPLICIT Null,
        listofEvents                  [2] IMPLICIT SEQUENCE OF Gn_AccessSpecification
    },
    additionalInfoRequested           [3] IMPLICIT BitString8 {
        recoveredMessageRequested         (0),  --true for requested
        objectRevisionRequested           (1)  --true for requested
}

QueryEventSummaryList-ResponsePDU ::= CHOICE {
    listofEvents                      [1] IMPLICIT SEQUENCE {
        listofAccessResult            [0] IMPLICIT BitString,
        listofEventMessage                Ev_ListOfEventResponse,
        listofObjectRevision          [5]  IMPLICIT SEQUENCE OF Gn_ObjectRevision
    },
    allEvents                         [2 IMPLICIT SEQUENCE {
        listofEventsID                    [0] IMPLICIT SEQUENCE OF Gn_AccessSpecification,
        listofEventMessage                Ev_ListOfEventResponse,
        listofObjectRevision          [5]  IMPLICIT SEQUENCE OF Gn_ObjectRevision
    }
}
```

-- The listofAccessResult parameter specifies for each event requested if the access was successful (bit=1) or not (bit=0).

4.3.15 Type Definitions

4.3.15.1 AP ASE Types

4.3.15.1.1 Ap_AccessProtectionSupported

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

4.3.15.1.2 Ap_AccessControl

```
Ap_AccessControl ::= BitString {            -- The Password (Unsigned8) is encoded as a bit string.
    password_Bit1                           (7),
    password_Bit2                           (6),
    password_Bit3                           (5),
    password_Bit4                           (4),
    password_Bit5                           (3),
    password_Bit6                           (2),
    password_Bit7                           (1),
    password_Bit8                           (0),
    access_Groups-1                             (15),
    access_Groups-2                             (14),
    access_Groups-3                             (13),
    access_Groups-4                             (12),
    access_Groups-5                             (11),
    access_Groups-6                             (10),
    access_Groups-7                             (9),
    access_Groups-8                             (8)
}
```

4.3.15.2 AR ASE Types

4.3.15.2.1 Ar_Dedicated

```
Ar_Dedicated  ::=  Boolean                  -- The value of TRUE means the AREP is dedicated.
                                            -- The value of FALSE means the AREP is not dedicated.
```

4.3.15.2.2 AREP

```
RespondingAREP ::= Unsigned32               -- Responding DLCEP address

RequestingAREP ::= Unsigned32               -- Requesting DLCEP address
```

4.3.15.2.3 Abort Types

4.3.15.2.3.1. Identifier

```
Identifier ::= Unsigned8 {
      user                                    (1),
      nma                                     (5)
}
```

4.3.15.2.3.2. Reason Code

ReasonCode ::= Unsigned8

4.3.15.2.3.3. Additional Detail

AdditionalDetaill ::= OctetString

4.3.15.2.4 Ar_ServiceTimeOut

Ar_ServiceTimeOut ::= Unsigned8

4.3.15.3 Data Link Layer Types

4.3.15.3.1 Dl_CyclicInterval

```
Dl_CyclicInterval  ::=  Unsigned32          -- Granularity is defined in IEC 61158-3 and IEC 61158-4.
```

4.3.15.3.2 Dl_DlsapAddress

```
Dl_DlsapAddress ::= Unsigned32              -- The format of this type is defined in IEC 61158-4.
```

4.3.15.3.3 Dl_PermissibleJitter

```
Dl_PermissibleJitter ::=  Unsigned16        -- Granularity is defined in IEC 61158-3 and IEC 61158-4.
```

4.3.15.3.4 Dl_SduSize

Dl_SduSize   ::= Unsigned16

4.3.15.4 Data Type ASE Types

4.3.15.4.1 Dt_ListOfFields

```
Dt_ListOfFields ::= SEQUENCE {
      fieldDataType                     [0] IMPLICIT Gn_NumericID,
      optionalParametersMap             [2] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      fieldName                         [0] IMPLICIT Gn_Name OPTIONAL                    -- bit 1
}
```

4.3.15.5 Event ASE Types

4.3.15.5.1 Ev_AcknowledgmentDataSpecifier

Ev_AcknowledgmentDataSpecifier ::= Ev_DataSpecifier

4.3.15.5.2 Ev_AcknowledgementPolicy

```
Ev_AcknowledgementPolicy ::= Boolean        -- True means all messages are to be acknowledged.
                                            -- False means only the specified message is to be acknowledged.
```

4.3.15.5.3 Ev_ContainedEventData

```
Ev_ContainedEventData ::= Boolean           -- True means the event message contains event data.
                                            -- False means the event message does not contain event data.
```

4.3.15.5.4 Ev_ContainedMessageType

```
Ev_ContainedMessageType ::= Unsigned8 {
      simple                            (1),
      reportedEventCount                (2),
      eventDetectionTime                (3),
      composite                         (4),
      any                               (5)
}
```

4.3.15.5.5 Ev_DataSpecifier

```
Ev_DataSpecifier ::= CHOICE {
      variableID                              [0] IMPLICIT Gn_NumericID,
      dataTypeId                              [1] IMPLICIT Gn_NumericID,
      dataLength                              [2] IMPLICIT Unsigned8
}
```

4.3.15.5.6 Ev_EventCount

```
En_EventCount ::= Unsigned8
```

4.3.15.5.7 Ev_EventData

```
Ev_EventData  ::=  ANY
```

-- Ev_EventData contains the data associated with the respective event. If the data is not to be included in the notification, its length is 0.

4.3.15.5.8 Ev_EventMessage

```
Ev_EventMessage ::= CHOICE  {
      eventKeyAttributeOnly                   [0] IMPLICIT Gn_KeyAttribute,
      eventKeyAttributeWithCount              [1] IMPLICIT Ev_KeyAttributeWithMessageCount,
      eventKeyAttributeWithTime               [2] IMPLICIT Ev_KeyAttributeWithData,
      eventKeyAttributeWithData               [3] IMPLICIT Ev_KeyAttributeWithData
}

Ev_ListOfEventResponse :: = CHOICE {
      simple                                  [1] IMPLICIT SEQUENCE OF Ev_Status,
      recoveredCount                          [2] IMPLICIT SEQUENCE OF ANY,          --Ev_Status, RecoveredCount
      recoveredDetectionTime                  [3] IMPLICIT SEQUENCE OF ANY,          --Ev_tStatus,
RecoveredDetectionTime
      composite                               [4] IMPLICIT SEQUENCE OF ANY
                                                          --Ev_Status, RecoveredCount,

RecoveredDetectionTime
}
```

-- The components of Ev_Status & RecoveredCount shall be appended together without any gap.

-- The components of Ev_Status & RecoveredDetectionTime shall be appended together without any gap.

-- The components of Ev_Status & RecoveredCount & RecoveredDetectionTime shall be appended together without any gap.

```
Ev_Status ::= BitString8 {
      active                                  (0),  --true for active
      detected                                (1),  --true for detected
      enabled                                 (2),  --true for enabled
      acknowledged                            (3)   --true for acked
}

RecoveredCount ::= Unsigned8

RecoveredDetectionTime ::= Ev_TimeTag
```

4.3.15.5.9 Ev_EventStatus

```
Ev_EventStatus ::= SEQUENCE {
      active                                  [0] IMPLICIT Boolean,      -- True means active
      detected                                [1] IMPLICIT Boolean,      -- True means detected
      enabled                                 [2] IMPLICIT Boolean,      -- True means enabled
      acknowledgmentSupported                 [3] IMPLICIT Boolean,      -- True means supported
      acknowledged                            [4] IMPLICIT Boolean       -- True means acknowledged
}
```

4.3.15.5.10 Ev_ErrorStatus

```
Ev_ErrorStatus ::= Boolean                    -- True means the underlying condition is active.
                                              -- False means the underlying condition is not active.
```

4.3.15.5.11 Ev_LastReportedEventInfo

```
Ev_LastReportedEventInfo ::= SEQUENCE {
      optionalParametersMap           [0] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      count                           [1] IMPLICIT Unsigned8 OPTIONAL,              -- bit 1
      detectionTime                   [2] IMPLICIT Ev_TimeTag OPTIONAL              -- bit 2
}
```

4.3.15.5.12 Ev_MessageType

```
Ev_MessageType ::= Unsigned8 {
      simpleMessage                   (1),
      reportedEventCount              (2),
      eventDetectionTime              (3),
      composite                       (4),
      simpleMessageWithData           (5),
      reportedEventCountWithData      (6),
      eventDetectionTimeWithData      (7),
      compositeWithData               (8)
}
```

4.3.15.5.13 Ev_NotificationType

```
Ev_NotificationType ::= Unsigned8 {
      basic                           (1),
      sequenced                       (2),
      timeOfNotification              (3),
      compound                        (4)
}
```

4.3.15.5.14 Ev_NumberToRecover

```
Ev_NumberToRecover  ::=  Unsigned8
```

4.3.15.5.15 Ev_SequenceNumber

```
Ev_SequenceNumber ::= Unsigned8
```

4.3.15.5.16 Ev_SelectionCriterion

```
Ev_SelectionCriterion ::= SEQUENCE {
      enabled                         [0] IMPLICIT Boolean,
      detected                        [1] IMPLICIT Boolean,
      acked                           [2] IMPLICIT Boolean,
      active                          [3] IMPLICIT Boolean
}
```

4.3.15.5.17 Ev_TimeTag

```
Ev_TimeTag  ::= Unsigned16
```

4.3.15.6 Function Invocation ASE Types

4.3.15.6.1 Fi_ActionInvokeID

```
Fi_ActionInvokeID ::= Unsigned8
```

4.3.15.6.2 Fi_AccessPrivilege

```
Fi_AccessPrivilege ::= BitString {
    rightToStartPassword           (23),
    rightToStopPassword            (22),
    rightToDeletePassword          (21),
    rightToStartAccessGroup        (19),
    rightToStopAccessGroup         (18),
    rightToDeleteAccessGroup       (17),
    rightToStartAllPartner         (31),
    rightToStopAllPartner          (30),
    rightToDeleteAllPartner        (29),
    password_Bit1                  (7),    -- The Password (Unsigned8) is encoded as a bit string.
    password_Bit2                  (6),
    password_Bit3                  (5),
    password_Bit4                  (4),
    password_Bit5                  (3),
    password_Bit6                  (2),
    password_Bit7                  (1),
    password_Bit8                  (0),
    access_Groups-1                    (15),
    access_Groups-2                    (14),
    access_Groups-3                    (13),
    access_Groups-4                    (12),
    access_Groups-5                    (11),
    access_Groups-6                    (10),
    access_Groups-7                    (9),
    access_Groups-8                    (8)
}
```

4.3.15.6.3 Fi_SegmentNumber

```
Fi_SegmentNumber ::= Unsigned32        -- The first segment number shall be one(1).
```

4.3.15.6.4 Fi_State

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

4.3.15.7 General Types

4.3.15.7.1 Gn_AccessSpecification

```
Gn_AccessSpecification ::= CHOICE {
    index                          [0] IMPLICIT Index,
    name                           [1] IMPLICIT Name
}
```

4.3.15.7.2 Gn_Deletable

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

4.3.15.7.3 Gn_Enabled

```
Gn_Enabled ::= Boolean             -- True means enabled.
                                   -- False means disabled.
```

4.3.15.7.4 Gn_FullyNestedTypeDescription

```
Gn_FullyNestedTypeDescription  ::=  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,
      float32                                [8]  Gn_Length,
      float64                                [15]  Gn_Length,
      binaryDate                             [11]  Gn_Length,
      timeOfDay                              [12]  Gn_Length,
      timeDifference                         [13]  Gn_Length,
      universalTime                          [16]  Gn_Length,
      fieldbusTime                           [17]  Gn_Length,
      time                                   [21]  Gn_Length,
      bitstring8                             [22]  Gn_Length,
      bitstring16                            [23]  Gn_Length,
      bitstring32                            [24]  Gn_Length,
      visiblestring1                         [25]  Gn_Length,
      visiblestring2                         [26]  Gn_Length,
      visiblestring4                         [27]  Gn_Length,
      visiblestring8                         [28]  Gn_Length,
      visiblestring16                        [29]  Gn_Length,
      octetstring1                           [30]  Gn_Length,
      octetstring2                           [31]  Gn_Length,
      octetstring4                           [32]  Gn_Length,
      octetstring8                           [33]  Gn_Length,
      octetstring16                          [34]  Gn_Length,
      bcd                                    [35]  Gn_Length,
      iso10646char                           [36]  Gn_Length,
      binarytime0                            [40]  Gn_Length,
      binarytime1                            [41]  Gn_Length,
      binarytime2                            [42]  Gn_Length,
      binarytime3                            [43]  Gn_Length,
      binarytime4                            [44]  Gn_Length,
      binarytime5                            [45]  Gn_Length,
      binarytime6                            [46]  Gn_Length,
      binarytime7                            [47]  Gn_Length,
      binarytime8                            [48]  Gn_Length,
      binarytime9                            [49]  Gn_Length,
      visiblestring                          [9]  Gn_Length,
      octetstring                            [10]  Gn_Length,
      bitstring                              [14]  Gn_Length,
      compactBooleanArray                    [37]  Gn_Length,
      compactBCDArray                        [38]  Gn_Length,
      iso646string                           [39]  Gn_Length,
      array                                  [55]  IMPLICIT SEQUENCE {
            number-of-elements                     [0]  IMPLICIT Unsigned8,
            element-type                                [1]  Gn_FullyNestedTypeDescription
      },
      structure                                    [56]  IMPLICIT SEQUENCE OF Gn_FullyNestedTypeDescription
}
```

4.3.15.7.5 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
}
```

4.3.15.7.6 Gn_Length

```
Gn_Length ::= Unsigned8
```

4.3.15.7.7 Gn_ListOfVariableAccess

```
Gn_ListOfVariableAccess  ::=  SEQUENCE OF CHOICE  {
      numericID                                    [0] IMPLICIT Gn_NumericID,
      name                                         [1] IMPLICIT Gn_Name,
      listName                                     [2] IMPLICIT Gn_Name,
      symbolicAddress                              [5] IMPLICIT Gn_SymbolicAddress,
      partialNumericID                             [6] IMPLICIT SEQUENCE {
           numericID                               [0] IMPLICIT Gn_NumericID,
           partialAccess                                     Gn_PartialAccess
      },
      partialName                                  [7] IMPLICIT SEQUENCE {
           name                                         [0] IMPLICIT Gn_Name,
           partialAccess                                          Gn_PartialAccess
      },
      partialListName                              [8] IMPLICIT SEQUENCE {
           listName                                [0] IMPLICIT Gn_Name,
           partialAccess                                          Gn_PartialAccess
      }
}
```

4.3.15.7.8 Gn_MoreFollows

```
Gn_MoreFollows ::= Boolean
```

4.3.15.7.9 Gn_Name

```
Gn_Name  ::=  OctetString
```

4.3.15.7.10 Gn_NameWithComponent

```
Gn_NameWithComponent  ::= SEQUENCE  {
      name                                         [0] IMPLICIT Gn_Name,
      component                                    [1] IMPLICIT Unsigned16
}
```

4.3.15.7.11 Gn_NumericAddress

```
Gn_NumericAddress  ::=  SEQUENCE  {
      startAddress                                 [0] IMPLICIT Unsigned32,   -- physical address of the starting location
      length                                       [1] IMPLICIT Unsigned16    -- octet length of a memory block
}
```

4.3.15.7.12 Gn_NumericID

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

4.3.15.7.13 Gn_NumericIdWithComponent

```
Gn_NumericIdWithComponent  ::= SEQUENCE  {
      numericID                                    [0] IMPLICIT Gn_NumericID,
      component                                    [1] IMPLICIT Unsigned16
}
```

4.3.15.7.14 Gn_ObjectDefinition

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

4.3.15.7.15 Gn_ObjectRevision

```
Gn_ObjectRevision  ::= Unsigned8
      -- bits 8 - 5 are used as the Major Revision
      -- bits 4 -1 are used as the Minor Revision
      -- each may have a value between zero and 15 inclusive
```

4.3.15.7.16 Gn_OptionalParametersMaps

```
Gn_OptionalParametersMap8  ::=  BitString8

Gn_OptionalParametersMap16  ::=  BitString16

Gn_OptionalParametersMap32  ::=  BitString32
```

```
--  The above types have special semantics to represent which optional parameters are included
--  in the transfer syntax. Bit 1 corresponds to the first optional parameter of a given primitive,
--  bit 2 corresponds to the second optional parameter, up to bit "n" corresponds to the "n-th"
--  optional parameters. The value of one indicates that the corresponding optional parameter is
--  present in the transfer syntax, and the value of zero indicates that it is not present.
```

4.3.15.7.17 Gn_PartialAccess

```
Gn_PartialAccess  ::=  CHOICE  {
    componentID                     [3] IMPLICIT Gn_SubIndex,
    componentList                   [20] IMPLICIT SEQUENCE {
        nestingLevel                        [1] Unsigned8,
        accessedElements                    [2] SEQUENCE OF Gn_Sub-Index
    }
}
```

-- The componentID of the partial access selects a single element on the first level of nesting for access. The componentList specifies one possibly complex base element on a specified level of nesting and, if necessary, selects a number of subelements from this base element. The first elements (their number is given by nestingLevel) of the accessedElements list define the path to the base element. The rest of the list selects a number of subelements to be accessed. If the rest of the list is empty, the complete base element is accessed.

4.3.15.7.18 Gn_PhysicalAddress

```
Gn_PhysicalAddress ::= Unsigned32
```

4.3.15.7.19 Gn_RequestFlag

```
Gn_RequestFlag  ::=  Boolean        -- TRUE means the specified parameter is requested in the response.
                                    -- FALSE means the specified parameter is not requested in the response.
```

4.3.15.7.20 Gn_Reusable

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

4.3.15.7.21 Gn_SubIndex

```
Gn_SubIndex ::= Unsigned8
```

4.3.15.7.22 Gn_SymbolicAddress

```
Gn_SymbolicAddress  ::=  VisibleString
```

4.3.15.7.23 Gn_TypeDescription

```
Gn_TypeDescription  ::= SEQUENCE OF CHOICE {
    simple                          [1]  IMPLICIT ANY,
        -- Octet1: High byte Data Type Gn_NumericID
        -- Octet2: Low byte Data Type Gn_NumericID
        -- Octet3: Length
    array                           [2]  IMPLICIT ANY,
        -- Octet1: High byte Data Type Gn_NumericID
        -- Octet2: Low byte Data Type Gn_NumericID
        -- Octet3: Length
        -- Octet4: Number of Elements
    structure                       [3]  IMPLICIT ANY
        -- Octet 1    : High byte Data Type Index (Element 1)
        -- Octet 2    : Low byte Data Type Index (Element 1)
        -- Octet 3    : Length (Element 1)
        -- Octet 3n-2 : High byte Data Type Index (Element n)
        -- Octet 3n-1 : Low byte Data Type Index (Element n)
        -- Octet 3n-0 : Length (Element n)
 }
```

4.3.15.7.24 Gn_ObjectClass

```
Gn_ObjectClass ::= Unsigned8 {
    anyObject                          (0),
    loadRegion                         (2),
    functionInvocation                 (3),
    event                              (4),
    fixedLength&StringDataType         (5),
    structuredDataType                 (6),
    fixedLength&StringVariable         (7),
    arrayVariable                      (8),
    dataStructureVariable              (9),
    variableList                       (10),
    arrayDataType                      (12),
    eventList                          (17),
    notifier                           (18),
    action                             (19)
}
```

4.3.15.8 Load Region ASE Types

4.3.15.8.1 Lr_FixedLengthSegment

```
Lr_FixedLengthSegment ::= Boolean      -- True means LR supports fixed-length segments only.
                                       -- False means LR supports variable-length segments.
```

4.3.15.8.2 Lr_LocalAddress

```
Lr_LocalAddress ::= Unsigned32
```

4.3.15.8.3 Lr_Size

```
Lr_Size ::= Unsigned32
```

4.3.15.8.4 Lr_State

```
Lr_State ::= Unsigned8 {
    NotDownLoadable                    (1),
    downLoadable                       (2),
    downLoading                        (3),
    downloadFailure                    (4),
    downloadSuccess                    (5),
    loaded                             (6),
    inUse                              (7)
}
```

4.3.15.8.5 Lr_TerminateReason

```
Lr_TerminateReason ::= Boolean         -- True means the load process was successfully completed.
                                       -- False means the load process was completed with failure.
```

4.3.15.8.6 Lr_UploadState

```
Lr_UploadState ::= Unsigned8 {
    notUploadable                      (1),
    uploadable                         (2),
    uploading                          (3),
    completingUpload                   (4)
}
```

4.3.15.9 Management ASE Types

4.3.15.9.1 Attribute Maps

4.3.15.9.1.1. Mn_ActionAttrMap

```
Mn_ActionAttrMap  ::=  Bitstring16  {
      commonAttrValues                              (1),
      listOfSupportedAttributes                     (2),
      listOfOperationalServices                     (3),
      reusable                                      (4),
      listOfRelatedObjects                          (5),
      actionInvokeServiceArgumentDataType           (6),
      actionReturnServiceArgumentDataType           (7)
}
```

4.3.15.9.1.2. Mn_ApAttrMap

```
Mn_ApAttrMap ::= BitString16  {
      commonAttrValues                              (1),
      listOfSupportedAttributes                     (2),
      listOfSupportedServices                       (3),
      manufacturerIdentifier                        (4),
      ApdescriptorReference                         (5),
      networkAccessState                            (6),
      listOfArEndpointInfo                          (7),
      listOfInuseArEndpointInfo                     (8),
      listOfSupportedApoClassAndObjects             (9),
      listOfDlsapAddresses                          (10),
      listOfEventSummarySelectionCriteria           (11)
}
```

4.3.15.9.1.3. Mn_AttributeMap

```
Mn_AttributeMap  ::=  SEQUENCE {
      Mn_CommonAttrMap,
      attributeMap  CHOICE  {
           apAttrMap                                [1] IMPLICIT Mn_ApAttrMap,
           dataTypeAttrMap                          [2] IMPLICIT Mn_DataTypeAttrMap,
           variableAttrMap                          [3] IMPLICIT Mn_VariableAttrMap,
           variableListAttrMap                      [4] IMPLICIT Mn_VariableListAttrMap,
           eventAttrMap                             [5] IMPLICIT Mn_EventAttrMap,
           eventListAttrMap                         [6] IMPLICIT Mn_EventListAttrMap,
           notifierAttrMap                          [7] IMPLICIT Mn_NotifierAttrMap,
           loadRegionAttrMap                        [8] IMPLICIT Mn_LoadRegionAttrMap,
           functionInvocationAttrMap                [9] IMPLICIT Mn_FunctionInvocationAttrMap,
           actionAttrMap                            [10] IMPLICIT Mn_ActionAttrMap
      }
}
```

4.3.15.9.1.4. Mn_CommonAttrMap

```
Mn_CommonAttrMap  ::=  BitString8  {
      numericID                                     (1),
      name                                          (2),
      instanceOf                                    (3),
      parentClass                                   (4),
      userDescription                               (5),
      objectRevision                                (6)
}
```

4.3.15.9.1.5. Mn_DataTypeAttrMap

```
Mn_DataTypeAttrMap  ::=  BitString16 {
      commonAttrValues                        (1),
      listOfSupportedAttributes               (2),
      dataTypeNumericId                       (3),
      dataTypeName                            (4),
      format                                  (5),
      octetLength                             (6),
      numberOfFields                          (7),
      listOfFields                            (8),
      numberOfArrayElements                   (9),
      arrayElementDataType                    (10)
}
```

4.3.15.9.1.6. Mn_EventAttrMap

```
Mn_EventAttrMap  ::=  BitString16 {
      commonAttrValues                        (1),
      listOfSupportedAttributes               (2),
      listOfOperationalServices               (3),
      eventStatus                             (4),
      messageType                             (5),
      accessPrivilege                         (6),
      lastReportedEventInfo                   (7),
      eventDataSpecifier                      (8),
      acknowledgmentDataSpecifier             (9)
}
```

4.3.15.9.1.7. Mn_EventListAttrMap

```
Mn_EventListAttrMap  ::=  BitString8  {
      commonAttrValues                        (1),
      listOfSupportedAttributes               (2),
      listOfOperationalServices               (3),
      numberOfEntries                         (4),
      listOfEvents                            (5)
}
```

4.3.15.9.1.8. Mn_FunctionInvocationAttrMap

```
Mn_FunctionInvocationAttrMap  ::=  BitString16  {
      commonAttrValues                        (1),
      listOfSupportedAttributes               (2),
      listOfOperationalServices               (3),
      accessPrivilege                         (4),
      deletable                               (5),
      reusable                                (6),
      functionInvocationState                 (7),
      numberOfRelatedObjectsInUse             (8),
      listOfRelatedObjects                    (9),
      startArgumentDataTypes                  (10),
      stopServiceArgumentDataTypes            (11),
      resumeServiceArgumentDataTypes          (12),
      resetServiceArgumentDataTypes           (13),
      killServiceArgumentDataTypes            (14),
      executionArgument                       (15)
}
```

4.3.15.9.1.9. Mn_LoadRegionAttrMap

```
Mn_LoadRegionAttrMap  ::=  BitString16  {
      commonAttrValues                          (1),
      listOfSupportedAttributes                 (2),
      listOfOperationalServices                 (3),
      loadRegionSize                            (4),
      accessPrivilege                           (5),
      localAddress                              (6),
      loadRegionState                           (7),
      uploadState                               (8),
      numberOfRelatedObjectsInUse               (9),
      listOfRelatedObjects                      (10),
      contentsSize                              (11),
      loadImageName                             (12),
      fixedLengthSegment                        (13),
      fixedSegmentLength                        (14),
      intersegmentRequestTimeout                (15),
      loadRegionSharable                        (16)
}
```

4.3.15.9.1.10. Mn_NotifierAttrMap

```
Mn_NotifierAttrMap  ::=  BitString16 {
      commonAttrValues                          (1),
      listOfSupportedAttributes                 (2),
      listOfOperationalServices                 (3),
      enabled                                   (4),
      notificationArepClass                     (5),
      notificationArep                          (6),
      notificationType                          (7),
      containedMessageType                      (8),
      containedEventData                        (9),
      lastNotificationSequenceNumber            (10),
      listOfEvents                              (11)
}
```

4.3.15.9.1.11. Mn_VariableAttrMap

```
Mn_VariableAttrMap  ::=  BitString16 {
      commonAttrValues                          (1),
      listOfSupportedAttributes                 (2),
      listOfOperationalServices                 (3),
      symbolicAddress                           (4),
      dataType                                  (5),
      length                                    (6),
      variableLengthConveyance                  (7),
      numberOfArrayElements                     (8),
      accessPrivilege                           (9),
      localDetail                               (10),
      fullyNestedTypeDescription                (11)
}
```

4.3.15.9.1.12. Mn_VariableListAttrMap

```
Mn_VariableListAttrMap  ::=  BitString8 {
      commonAttrValues                          (1),
      listOfSupportedAttributes                 (2),
      listOfOperationalServices                 (3),
      numberOfEntries                           (4),
      listOfVariables                           (5),
      deletable                                 (6),
      accessPrivilege                           (7)
}
```

4.3.15.9.2 Attribute Values

4.3.15.9.2.1. Mn_ActionAttrValues

```
Mn_ActionAttrValues  ::=  SEQUENCE  {
      optionalParametersMap                 [0] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
      commonAttrValues                      [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,             -- bit 1
      listOfSupportedAttributes             [2] IMPLICIT Mn_ActionAttrMap  OPTIONAL,                -- bit 2
      listOfOperationalServices             [3] IMPLICIT Mn_ActionServiceMap  OPTIONAL,             -- bit 3
      reusable                              [4] IMPLICIT Gn_Reusable DEFAULT TRUE,                  -- bit 4
      listOfRelatedObjects                  [5] IMPLICIT SEQUENCE OF Gn_KeyAttribute OPTIONAL,      -- bit 5
      actionInvokeServiceArgumentDataType   [6] IMPLICIT SEQUENCE OF Gn_NumericID OPTIONAL,         -- bit 6
      actionActionServiceArgumentDataType   [7] IMPLICIT SEQUENCE OF Gn_NumericID  OPTIONAL         -- bit 7
}
```

4.3.15.9.2.2. Mn_ApAttrValues

```
Mn_ApAttrValues  ::=  SEQUENCE {
      optionalParametersMap                 [0] IMPLICIT Gn_OptionalParametersMap16  OPTIONAL,
      commonAttrValues                      [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,                    -- bit 1
      listOfSupportedAttributes             [2] IMPLICIT Mn_ApAttrMap OPTIONAL,                            -- bit 2
      listOfSupportedServices               [3] IMPLICIT Mn_ApServiceMap OPTIONAL,                         -- bit 3
      manufacturerIdentifier                [4] IMPLICIT Mn_ManufacturerIdentifier OPTIONAL,               -- bit 4
      ApdescriptorReference                 [5] IMPLICIT OctetString OPTIONAL,                             -- bit 5
      networkAccessState                    [6] IMPLICIT Mn_NetworkAccessState OPTIONAL,                   -- bit 6
      listOfArEndpointInfo                  [7] IMPLICIT SEQUENCE OF Mn_ArEndpointInfo OPTIONAL,           -- bit 7
      listOfInuseArEndpointInfo             [8] IMPLICIT SEQUENCE OF Mn_InuseArEndpointInfo OPTIONAL,      -- bit 8
      listOfSupportedApoClassAndObjects     [9] IMPLICIT SEQUENCE OF Mn_ApoClassAndObjects OPTIONAL,       -- bit 9
      listOfDlsapAddresses                  [10] IMPLICIT SEQUENCE OF Dl_DlsapAddress OPTIONAL,            -- bit 10
      listOfEventSummarySelectionCriteria   [11] IMPLICIT SEQUENCE OF Ev_SelectionCriterion OPTIONAL       -- bit 11
}
```

4.3.15.9.2.3. Mn_AttributeValues

```
Mn_AttributeValues  ::=  CHOICE  {
      ApAttrValues                          [1] IMPLICIT Mn_ApAttrValues,
      dataTypeAttrValues                    [2] IMPLICIT Mn_DataTypeAttrValues,
      variableAttrValues                    [3] IMPLICIT Mn_VariableAttrValues,
      variableListAttrValues                [4] IMPLICIT Mn_VariableListAttrValues,
      eventAttrValues                       [5] IMPLICIT Mn_EventAttrValues,
      eventListAttrValues                   [6] IMPLICIT Mn_EventListAttrValues,
      notifierAttrValues                    [7] IMPLICIT Mn_NotifierAttrValues,
      loadRegionAttrValues                  [8] IMPLICIT Mn_LoadRegionAttrValues,
      functionInvocationAttrValues          [9] IMPLICIT Mn_FunctionInvocationAttrValues,
      actionAttrValues                      [10] IMPLICIT Mn_ActionAttrValues
}
```

4.3.15.9.2.4. Mn_CommonAttrValues

```
Mn_CommonAttrValues  ::=  SEQUENCE  {
      optionalParametersMap                 [0] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
      numericID                             [1] IMPLICIT Gn_NumericID,                   -- bit 1
      name                                  [2] IMPLICIT Gn_Name  OPTIONAL,              -- bit 2
      instanceOf                            [3] IMPLICIT Gn_NumericID  OPTIONAL,         -- bit 3
      parentClass                           [4] IMPLICIT Gn_NumericID  OPTIONAL,         -- bit 4
      userDescription                       [5] IMPLICIT OctetString OPTIONAL,           -- bit 5
      objectRevision                        [6] IMPLICIT Gn_ObjectRevision  OPTIONAL     -- bit 6
}
```

4.3.15.9.2.5. Mn_DataTypeAttrValues

```
Mn_DataTypeAttrValues  ::=  SEQUENCE {
    optionalParametersMap           [0] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
    commonAttrValues                [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,            -- bit 1
    listOfSupportedAttributes       [2] IMPLICIT Mn_DataTypeAttrMap  OPTIONAL,             -- bit 2
    dataTypeNumericId               [3] IMPLICIT Gn_NumericID OPTIONAL,                    -- bit 3
    dataTypeName                    [4] IMPLICIT Gn_Name OPTIONAL,                         -- bit 4
    format                          [5] IMPLICIT Gn_NumericID OPTIOINAL,                   -- bit 5
    octetLength                     [6] IMPLICIT Unsigned8  OPTIONAL,                      -- bit 6
    numberOfFields                  [7] IMPLICIT Unsigned8 OPTIONAL,                       -- bit 7
    listOfFields                    [8] IMPLICIT SEQUENCE OF Dt_ListOfFields  OPTIONAL     -- bit 8
    numberOfArrayElements           [9] IMPLICIT Gn_NumericID  OPTIONAL,                   -- bit 9
    arrayElementDataType            [10] IMPLICIT Gn_NumericID  OPTIONAL                   -- bit 10
}
```

4.3.15.9.2.6. Mn_EventAttrValues

```
Mn_EventAttrValues  ::=  SEQUENCE {
    optionalParametersMap           [0] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
    commonAttrValues                [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,            -- bit 1
    listOfSupportedAttributes       [2] IMPLICIT Mn_EventAttrMap  OPTIONAL,                -- bit 2
    listOfOperationalServices       [3] IMPLICIT Mn_EventServiceMap  OPTIONAL,             -- bit 3
    eventStatus                     [4] IMPLICIT Ev_EventStatus OPTIONAL,                  -- bit 4
    messageType                     [5] IMPLICIT Ev_MessageType  OPTIONAL,                 -- bit 5
    accessPrivilege                 [6] IMPLICIT Vr_VariablelistAccessPrivilege OPTIONAL,  -- bit 6
    lastReportedEventInfo           [7] IMPLICIT Ev_LastReportedEventInfo OPTIONAL,        -- bit 7
    eventDataSpecifier              [8] IMPLICIT Ev_DataSpecifier  OPTIONAL,               -- bit 8
    acknowledgmentDataSpecifier     [9] IMPLICIT EV_AcknowledgementDataSpecifier OPTIONAL  -- bit 9
}
```

4.3.15.9.2.7. Mn_EventListAttrValues

```
Mn_EventListAttrValues  ::=  SEQUENCE  {
    optionalParametersMap           [0] IMPLICIT Gn_OptionalParametersMap8 OPTIONAL,
    commonAttrValues                [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,            -- bit 1
    listOfSupportedAttributes       [2] IMPLICIT Mn_EventListAttrMap OPTIONAL,             -- bit 2
    listOfOperationalServices       [3] IMPLICIT Mn_EventListServiceMap OPTIONAL,          -- bit 3
    numberOfEntries                 [4] IMPLICIT Unsigned16 OPTIONAL,                      -- bit 4
    listOfEvents                    [5] IMPLICIT SEQUENCE OF Gn_NumericID OPTIONAL         -- bit 5
}
```

4.3.15.9.2.8. Mn_FunctionInvocationAttrValues

```
Mn_FunctionInvocationAttrValues  ::=  SEQUENCE  {
    optionalParametersMap           [0] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
    commonAttrValues                [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,            -- bit 1
    listOfSupportedAttributes       [2] IMPLICIT Mn_FunctionInvocationAttrMap  OPTIONAL,   -- bit 2
    listOfOperationalServices       [3] IMPLICIT Mn_FunctionInvocationServiceMap  OPTIONAL, -- bit 3
    accessPrivilege                 [4] IMPLICIT Vr_AccessPrivilege OPTIONAL,              -- bit 4
    deletable                       [5] IMPLICIT Mn_Deletable OPTIONAL,                    -- bit 5
    reusable                        [6] IMPLICIT Mn_Reusable DEFAULT TRUE OPTIONAL,        -- bit 6
    functionInvocationState         [7] IMPLICIT Fi_State  OPTIONAL,                       -- bit 7
    numberOfRelatedObjectsInUse     [8] IMPLICIT Unsigned8 OPTIONAL,                       -- bit 8
    listOfRelatedObjects            [9] IMPLICIT SEQUENCE OF Gn_KeyAttribute  OPTIONAL,    -- bit 9
    startArgumentDataTypes          [10] IMPLICIT Gn_NumericID OPTIONAL,                   -- bit 10
    stopServiceArgumentDataTypes    [11] IMPLICIT Gn_NumericID OPTIONAL,                   -- bit 11
    resumeServiceArgumentDataTypes  [12] IMPLICIT Gn_NumericID OPTIONAL,                   -- bit 12
    resetServiceArgumentDataTypes   [13] IMPLICIT Gn_NumericID OPTIONAL,                   -- bit 13
    killServiceArgumentDataTypes    [14] IMPLICIT Gn_NumericID OPTIONAL,                   -- bit 14
    executionArgument               [15] IMPLICIT OctetString OPTIONAL                     -- bit 15
}
```

4.3.15.9.2.9. Mn_LoadRegionAttrValues

```
Mn_LoadRegionAttrValues  ::=  SEQUENCE  {
     optionalParametersMap             [0] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
     commonAttrValues                  [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,              -- bit 1
     listOfSupportedAttributes         [2] IMPLICIT Mn_LoadRegionAttrValues  OPTIONAL,          -- bit 2
     listOfOperationalServices         [3] IMPLICIT Mn_LoadRegionServiceMap OPTIONAL,           -- bit 3
     loadRegionSize                    [4] IMPLICIT Lr_Size  OPTIONAL,                         -- bit 4
     accessPrivilege                   [5] IMPLICIT Vr_AccessPrivilege OPTIONAL,               -- bit 5
     localAddress                      [6] IMPLICIT Lr_LocalAddress OPTIONAL,                  -- bit 6
     loadRegionState                   [7] IMPLICIT Lr_State  OPTIONAL,                        -- bit 7
     uploadState                       [8] IMPLICIT Lr_UploadState  OPTIONAL,                  -- bit 8
     numberOfRelatedObjectsInUse       [9] IMPLICIT Unsigned8 OPTIONAL,                        -- bit 9
     listOfRelatedObjects              [10] IMPLICIT SEQUENCE OF Gn_AccessSpecification OPTIONAL, -- bit 10
     contentsSize                      [11] IMPLICIT Lr_Size  OPTIONAL,                        -- bit 11
     loadImageName                     [12] IMPLICIT Gn_Name  OPTIONAL,                        -- bit 12
     fixedLengthSegment                [13] IMPLICIT Lr_FixedLengthSegment OPTIONAL,            -- bit
13
     fixedSegmentLength                [14] IMPLICIT Lr_Size  OPTIONAL,                        -- bit 14
     intersegmentRequestTimeout        [15] IMPLICIT Ar_ServiceTimeOut  OPTIONAL                -- bit
15
     loadRegionSharable                [16] IMPLICIT Boolean OPTIONAL                          -- bit 16
}
```

4.3.15.9.2.10. Mn_NotifierAttrValues

```
Mn_NotifierAttrValues  ::=  SEQUENCE {
     optionalParametersMap             [0] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
     commonAttrValues                  [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,              -- bit 1
     listOfSupportedAttributes         [2] IMPLICIT Mn_NotifierAttrMap OPTIONAL,               -- bit 2
     listOfOperationalServices         [3] IMPLICIT Mn_NotifierServiceMap  OPTIONAL,            -- bit 3
     enabled                           [4] IMPLICIT Gn_Enabled  OPTIONAL,                      -- bit 4
     notificationArepClass             [5] IMPLICIT Gn_NumericID  OPTIONAL,                    -- bit 5
     notificationArep                  [6] IMPLICIT Gn_NumericID  OPTIONAL,                    -- bit 6
     notificationType                  [7] IMPLICIT Ev_NotificationType  OPTIONAL,              -- bit 7
     containedMessageType              [8] IMPLICIT Ev_ContainedMessageType  OPTIONAL,          -- bit 8
     containedEventData                [9] IMPLICIT Ev_ContainedEventData  OPTIONAL,            -- bit 9
     lastNotificationSequenceNumber    [10] IMPLICIT Ev_SequenceNumber  OPTIONAL,               -- bit
10
     listOfEvents                      [11] IMPLICIT Gn_NumericID OPTIONAL                     -- bit 11
}
```

4.3.15.9.2.11. Mn_VariableAttrValues

```
Mn_VariableAttrValues  ::=  SEQUENCE {
     optionalParametersMap             [0] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
     commonAttrValues                  [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,              -- bit 1
     listOfSupportedAttributes         [2] IMPLICIT Mn_VariableAttrMap  OPTIONAL,               -- bit 2
     listOfOperationalServices         [5] IMPLICIT Mn_VariableServiceMap  OPTIONAL,            -- bit 3
     symbolicAddress                   [4] IMPLICIT Gn_SymbolicAddress OPTIONAL,                -- bit 4
     dataType                          [5] IMPLICIT Gn_NumericID  OPTIONAL,                    -- bit 5
     length                            [6] IMPLICIT Unsigned8 OPTIONAL,                        -- bit 6
     variableLengthConveyance          [7] IMPLICIT Vr_variableLengthConveyance OPTIONAL,      -- bit 7
     numberOfArrayElements             [8] IMPLICIT Unsigned8 OPTIONAL,                        -- bit 8
     accessPrivilege                   [9] IMPLICIT Vr_AccessPrivilege OPTIONAL,               -- bit 9
     localAddress                      [10] IMPLICIT Gn_NumericAddress OPTIONAL                 -- bit
10
     fullyNestedTypeDescription        [11] IMPLICIT Gn_FullyNestedTypeDescription OPTIONAL     -- bit 11
}
```

4.3.15.9.2.12. Mn_VariableListAttrValues

```
Mn_VariableListAttrValues  ::=  SEQUENCE {
    optionalParametersMap         [0] IMPLICIT Gn_OptionalParametersMap8,
    commonAttrValues              [1] IMPLICIT Mn_CommonAttrValues  OPTIONAL,            -- bit 1
    listOfSupportedAttributes     [2] IMPLICIT Mn_VariableListAttrMap  OPTIONAL,         -- bit 2
    listOfOperationalServices     [3] IMPLICIT Mn_VariableListServiceMap  OPTIONAL,      -- bit 3
    numberOfEntries               [4] IMPLICIT Unsigned8 OPTIONAL,                       -- bit 4
    listOfVariables               [5] IMPLICIT SEQUENCE OF Gn_AccessSpecification OPTIONAL, -- bit 5
    deletable                     [6] IMPLICIT Gn_Deletable OPTIONAL,                    -- bit 6
    accessPrivilege               [7] IMPLICIT Vr_AccessPrivilege OPTIONAL               -- bit 7
}
```

4.3.15.9.3 Mn_ApoClassAndObjects

```
Mn_ApoClassAndObjects ::= SEQUENCE {
    numericIdentifier             [0] IMPLICIT Gn_NumericID,
    romRamFlag                    [1] IMPLICIT Gn_RomRamFlag,
    maxNameLength                 [2] IMPLICIT Unsigned8,
    accessProtectionSupported     [3] IMPLICIT Gn_AccessProtection,
    versionOfList                 [4] IMPLICIT Mn_Version,
    localReference                [5] IMPLICIT Unsigned32,
    optionalParametersMap         [6] IMPLICIT Gn_OptionalParametersMap16 OPTIONAL,
    numericIdentfier              [7] IMPLICIT Gn_NumericID OPTIONAL,          -- bit 1
    numberOfObjects               [8] IMPLICIT Unsigned8 OPTIONAL,             -- bit 2
    localReference                [9] IMPLICIT Unsigned32 OPTIONAL,            -- bit 3
    numericIdentfier              [10] IMPLICIT Gn_NumericID OPTIONAL,         -- bit 4
    numberOfObjects               [11] IMPLICIT Unsigned8 OPTIONAL,            -- bit 5
    localReference                [12] IMPLICIT Unsigned32 OPTIONAL,           -- bit 6
    numericIdentfier              [13] IMPLICIT Gn_NumericID OPTIONAL,         -- bit 7
    numberOfObjects               [14] IMPLICIT Unsigned8 OPTIONAL,            -- bit 8
    localReference                [15] IMPLICIT Unsigned32 OPTIONAL,           -- bit 9
    numericIdentfier              [16] IMPLICIT Gn_NumericID OPTIONAL,         -- bit 10
    numberOfObjects               [17] IMPLICIT Unsigned8 OPTIONAL,            -- bit 11
    localReference                [18] IMPLICIT Unsigned32 OPTIONAL            -- bit 12
}
```

4.3.15.9.4 Mn_ArEndpointInfo

```
Mn_ArEndpointInfo ::= SEQUENCE {
    configuredMaxPduSizeSending                    [0] IMPLICIT Unsigned8,
    configuredMaxPduSizeReceiving                  [1] IMPLICIT Unsigned8,
    configuredMaxOutstandingServicesRequesting     [2] IMPLICIT Unsigned8,
    configuredMaxOutstandingServicesResponding     [3] IMPLICIT Unsigned8,
    listOfSupportedServices                        [4] IMPLICIT Mn_ServiceMap,
    configuredMaxDataStructureNestingLevel         [5] IMPLICIT Unsigned8
}
```

4.3.15.9.5 Mn_Deletable

```
Mn_Deletable  ::=  Boolean                 -- The value of TRUE means the object is remotely deletable.
                                           -- The value of FALSE means the object is not remotely deletable.
```

4.3.15.9.6 Mn_InuseArEndpointInfo

```
Mn_InuseArEndpointInfo ::= SEQUENCE {
    negotiatedMaxPduSizeSending                    [0] IMPLICIT Unsigned8,
    negotiatedMaxPduSizeReceiving                  [1] IMPLICIT Unsigned8,
    negotiatedMaxOutstandingServicesRequesting     [2] IMPLICIT Unsigned8,
    negotiatedMaxOutstandingServicesResponding     [3] IMPLICIT Unsigned8,
    outstandingServicesRequestingCounter           [4] IMPLICIT Unsigned8,
    outstandingServicesRespondingCounter           [5] IMPLICIT Unsigned8,
    negotiatedMaxDataStructureNestingLevel   [     6] IMPLICIT Unsigned8
}
```

4.3.15.9.7 Mn_ManufacturerIdentifier

```
Mn_ManufacturerIdentifier ::= SEQUENCE {
    vendorName                          [0] IMPLICIT OctetString,
    modelIdentifier                     [1] IMPLICIT OctetString,
    vendorRevision                      [2] IMPLICIT OctetString,
    serialNumber                        [3] IMPLICIT OctetString
}
```

4.3.15.9.8 Mn_NetworkAccessState

```
Mn_NetworkAccessState ::= SEQUENCE {
    serviceAccessStatus                 [0] IMPLICIT Unsigned8 {
        readyForCommunication               (1),
        limitedNumberOfServices             (2),
        loadingNonInteracting               (3),
        loadingInteracting                  (4)
    },
    operationalStatus                       [1] IMPLICIT Unsigned8 {
        operational                         (1),
        partiallyOperational                (2),
        notOperational                      (3),
        needsMaintenance                    (4)
    }
}
```

4.3.15.9.9 Mn_PduSupportedMap

```
Mn_PduSupportedMap ::= BitString {
      getAttributes2PDU                      (1),      --All mandatory
      unsolicitedStatusPDU                (2),
      setAttributes2PDU                   (3),
      clientPulledDownloadPDU             (4),
      clientPullUploadPDU                 (5),
      serverPullDownloadPDU               (6),
      serverPulledUploadPDU               (7),
      createFunctionInvocationPDU         (8),
--    deleteFunctionInvocationPDU         (8),
      startPDU                            (9),
--    stopPDU                             (9),
--    resumePDU                           (9),
--    resetPDU                            (9),
      kill PDU                            (10),
      read2PDU                            (11),
      writePDU                            (12),
      physicalReadPDU                     (15),
      physicalWritePDU                    (16),
      informationReportPDU                (17),
      createVariableListPDU               (19),
--    deleteVariableListPDU               (19),
      eventNotificationPDU                (20),
      confirmedAcknowledgeEventPDU        (22),
      enableEventPDU                      (23),
      clientPushDownloadPDU               (25),
      getAttributesPDU                    (26),      -- Selective
      getAttributesListPDU                (27),
      subscribePDU                        (28),
      readListPDU                         (29),
      writeListPDU                        (30),
      informationReportListPDU            (31),
      exchangePDU                         (32),
      unconfirmedAcknowledgeEventsPDU     (33),
      getEventSummaryPDU                  (34),
      getEventSummaryListPDU              (35),
      queryEventSummaryPDU                (36),
      notificationRecoveryPDU             (37),
      discardPDU                          (38),
      actionInvokePDU                     (39),
      actionReturnPDU                     (40),
      setAttributes1PDU                   (41),
      read1PDU                            (42),
      getAttributes1PDU                   (43),
      concludePDU                         (44)
}
```

4.3.15.9.10 Mn_SelectedAttributes

```
Mn_SelectedAttributes ::= Unsigned8 {
      allMandatory                        (0),                -- All mandatory attributes
      userSpecified                       (128),              -- Specified by the AttributeMap
      minimumMandatory                    (255)               -- Minimum mandatory attributes
}
```

4.3.15.9.11 Mn_ServiceMap

4.3.15.9.11.1. Mn_ActionServiceMap

```
Mn_ActionServiceMap ::= BitString8 {
      actionInvoke                        (1),
      actionReturn                        (2)
}
```

4.3.15.9.11.2. Mn_ApServiceMap

```
Mn_ApServiceMap ::= BitString8 {
      subscribe                                     (1),
      identify                                      (2),
      getStatus                                     (3),
      statusNotification                            (4),
      initiate                                      (5),
      terminate                                     (6),
      conclude                                      (7)
}
```

4.3.15.9.11.3. Mn_EventServiceMap

```
Mn_EventServiceMap ::= BitString8 {
      unconfirmedAcknowledgeEvents                  (1),
      confirmedAcknowledgeEvents                    (2),
      enableEvents                                  (3),
      getEventSummary                               (4),
      getEventSummaryList                           (5),
      queryEventSummary                             (6)
}
```

4.3.15.9.11.4. Mn_EventListServiceMap

```
Mn_EventListServiceMap ::= BitString8 {
      getEventSummary                               (1),
      queryEventSummary                             (2)
}
```

4.3.15.9.11.5. Mn_FunctionInvocationServiceMap

```
Mn_FunctionInvocationServiceMap ::= BitString8 {
      start                                         (1),
      stop                                          (2),
      resume                                        (3),
      reset                                         (4),
      kill                                          (5)
}
```

4.3.15.9.11.6. Mn_LoadRegionServiceMap

```
Mn_LoadRegionServiceMap ::= BitString8 {
      initiateLoad                                  (1),
      pushSegment                                   (2),
      pullSegment                                   (3),
      terminateLoad                                 (4),
      discard                                       (5)
}
```

4.3.15.9.11.7. Mn_NotifierServiceMap

```
Mn_NotifierServiceMap ::= BitString8 {
      EventNotification                             (1),
      notificationRecovery                          (2)
}
```

4.3.15.9.11.8. Mn_VariableListServiceMap

```
Mn_VariableListServiceMap ::= BitString8 {
      read                                          (1),
      write                                         (2),
      readList                                      (3),
      writeList                                     (4),
      exchange                                      (5),
      informationReport                             (6),
      informationReportList                         (7)
}
```

4.3.15.9.11.9. Mn_VariableServiceMap

```
Mn_VariableListServiceMap ::= BitString8 {
      read                                    (1),
      write                                   (2),
      readList                                (3),
      writeList                               (4),
      exchange                                (5),
      informationReport                       (6),
      informationReportList                   (7)
}
```

4.3.15.10 Variable ASE Types

4.3.15.10.1 Vr_AccessPrivilege

```
Vr_AccessPrivilege ::= BitString {
      rightToReadPassword                     (23),
      rightToWritePassword                    (22),
      rightToDeletePassword                   (21),
      rightToReadAccessGroup                  (19),
      rightToWriteAccessGroup                 (18),
      rightToDeleteAccessGroup                (17),
      rightToReadAllPartners                  (31),
      rightToWriteAllPartners                 (30),
      rightToDeleteAllPartners                (29),
      password_Bit1                           (7),                -- The Password (Unsigned8) is encoded as a bit
string.
      password_Bit2                           (6),
      password_Bit3                           (5),
      password_Bit4                           (4),
      password_Bit5                           (3),
      password_Bit6                           (2),
      password_Bit7                           (1),
      password_Bit8                           (0),
      access_Groups-1                            (15),
      access_Groups-2                            (14),
      access_Groups-3                            (13),
      access_Groups-4                            (12),
      access_Groups-5                            (11),
      access_Groups-6                            (10),
      access_Groups-7                            (9),
      access_Groups-8                            (8)
}
```

4.3.15.10.2 Vr_VariableLengthConveyance

```
Vr_VariableLengthConveyance ::= Boolean   -- True means only the octets that are in use in a string are conveyed.
                                          -- False means all the octets in a string are conveyed.
```

4.3.15.10.3 Vr_WriteResult

```
Vr_WriteResult ::= CHOICE {
      success                                 [0] IMPLICIT NULL,
      service                                 [5] IMPLICIT Er_Service,
      access                                  [6] IMPLICIT Er_Access,
      others                                  [8] IMPLICIT Er_Other
}
```

4.4 Data Types

4.4.1 Notation for the Boolean Type

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

4.4.2 Notation for the Integer Type

Integer ::= INTEGER	-- any integer
Integer8 ::= INTEGER (-128..+127)	-- range -27 <= i <= 27-1
Integer16 ::= INTEGER (-32768..+32767)	-- range -2^{15} <= i <= 2^{15}-1
Integer32 ::= INTEGER	-- range -2^{31} <= i <= 2^{31}-1

4.4.3 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

4.4.4 Notation for the Floating Point Type

Floating32 ::= BIT STRING SIZE (4)	-- IEEE-754 Single precision
Floating64 ::= BIT STRING SIZE (8)	-- IEEE-754 Double precision

4.4.5 Notation for the BitString Type

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

4.4.6 Notation for the OctetString Type

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

4.4.7 Notation for VisibleString Type

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

4.4.8 Notation for the UNICODEString Type

UNICODEString ::= UNICODEString	-- 16-bit character code set defined in ISO 10646.

4.4.9 Notation for the FieldbusTime Type

FieldbusTime	-- Format and granularity are defined in IEC 61158-3 and IEC 61158-4.

4.4.10 Notation for the Universal Time Type

UniversalTime ::= VisibleString SIZE (12)	-- YYMMDDhhmmss

4.4.11 Notation for Binary Time Type

BinaryTime0	::= BIT STRING SIZE (16) --	10 µs resolution
BinaryTime1	::= BIT STRING SIZE (16) --	.1 ms resolution
BinaryTime2	::= BIT STRING SIZE (16) --	1 ms resolution
BinaryTime3	::= BIT STRING SIZE (16) --	10 ms resolution
BinaryTime4	::= BIT STRING SIZE (32) --	10 µs resolution
BinaryTime5	::= BIT STRING SIZE (32) --	.1 ms resolution
BinaryTime6	::= BIT STRING SIZE (32) --	1 ms resolution
BinaryTime7	::= BIT STRING SIZE (32) --	10 ms resolution
BinaryTime8	::= BIT STRING SIZE (48) --	10 µs resolution
BinaryTime9	::= BIT STRING SIZE (48) --	.1 ms resolution

4.4.12 Notation for BCD Type

BCD ::= Unsigned8 (0..9)	-- Lower four bits are used to express one BCD value.

4.4.13 Notation for Compact Boolean Array Type

```
CompactBooleanArray ::= BitString          -- Each zero bit representing Boolean value FALSE.
                                           -- Each one bit representing Boolean value TRUE.
                                           -- Unused bits, if any, shall be placed in bits 6-0 of the last octet.
```

4.4.14 Notation for Compact BCD Array Type

```
CompactBCDArray ::= OctetString            --  One BCD value is represented by four bits, an unused
                                           --  nibble, if any, shall be placed in bits 3-0 of the last octet,
                                           --  and shall be set to 1111F.
```

4.4.15 Notation for Date Type

Date ::= OctetString7

4.4.16 Notation for TimeOfDay Type

TimeOfDay ::= OctetString6

4.4.17 Notation for TimeDifference Type

TimeDifference ::= OctetString6

4.4.18 Notation for TimeValue Type

TimeValue ::= OctetString8

5 FAL Protocol State Machine

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

NOTE The state machines defined in this section and ARPMs defined in the annexes only define the valid events for each. It is a local matter to handle these invalid events.

The behavior of the FAL is described by three integrated protocol machines. Specific sets of these protocol machines are defined for different AREP types. The three protocol machines are: FAL Service Protocol Machine (FSPM), the Application Relationship Protocol Machine (ARPM), and the Data Link Layer Mapping Protocol Machine (DMPM). The relationship among these protocol machines as well as primitives exchanged among them are depicted in the following figure:

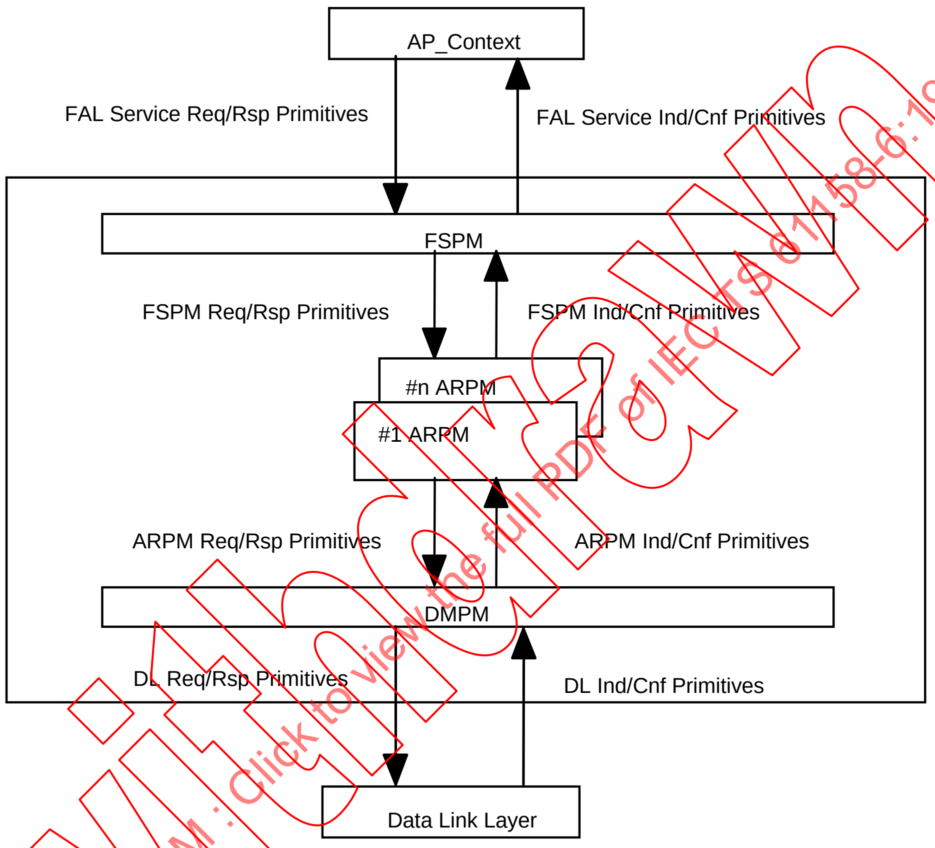


Figure 1 – Relationships among Protocol Machines and Adjacent Layers

- The FSPM describes the service interface between the AP_Context and a particular AREP. The FSPM is common to all the AREP classes and does not have any state changes. The FSPM is responsible for the following activities:
  a) to accept service primitives from the FAL service user and convert them into FAL internal primitives;
  b) to select an appropriate ARPM state machine based on the AREP Identifier parameter supplied by the AP_Context and send FAL internal primitives to the selected ARPM;
  c) to accept FAL internal primitives from the ARPM and convert them into service primitives for the AP_Context;
  d) to deliver the FAL service primitives to the AP_Context based on the AREP Identifier parameter associated with the primitives.
- The ARPM describes the establishment and release of an AR and exchange of FAL-PDUs with a remote ARPM(s). The ARPM is responsible for the following activities:

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

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

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

Since the ARPMs vary among different AREP types and new types may be added later, their definitions are provided in the normative annexes A to O of this technical specification.

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

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

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

5.1 AP Context State Machine

5.1.1 Primitive Definitions

5.1.1.1 Primitives Exchanged between FAL-User and AP-Context

Table 2 – Primitives issued by FAL-User to AP-Context

Primitive Name	Source	Associated Parameters and Functions
Terminate.req	FAL-User	Refer to FAL Service Definition (ISA-S50.02-5)
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 3 – Primitives issued by AP-Context to FAL-User

Primitive Name	Source	Associated Parameters and Functions
Terminate.ind	AP-Context	Refer to FAL Service Definition (ISA-S50.02-5)
Initiate.ind	AP-Context	
Initiate.cnf(+)	AP-Context	
Initiate.cnf(-)	AP-Context	
ConfirmedService.ind	AP-Context	
ConfirmedService.cnf	AP-Context	
UnconfirmedService.ind	AP-Context	

5.1.2 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*.

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"

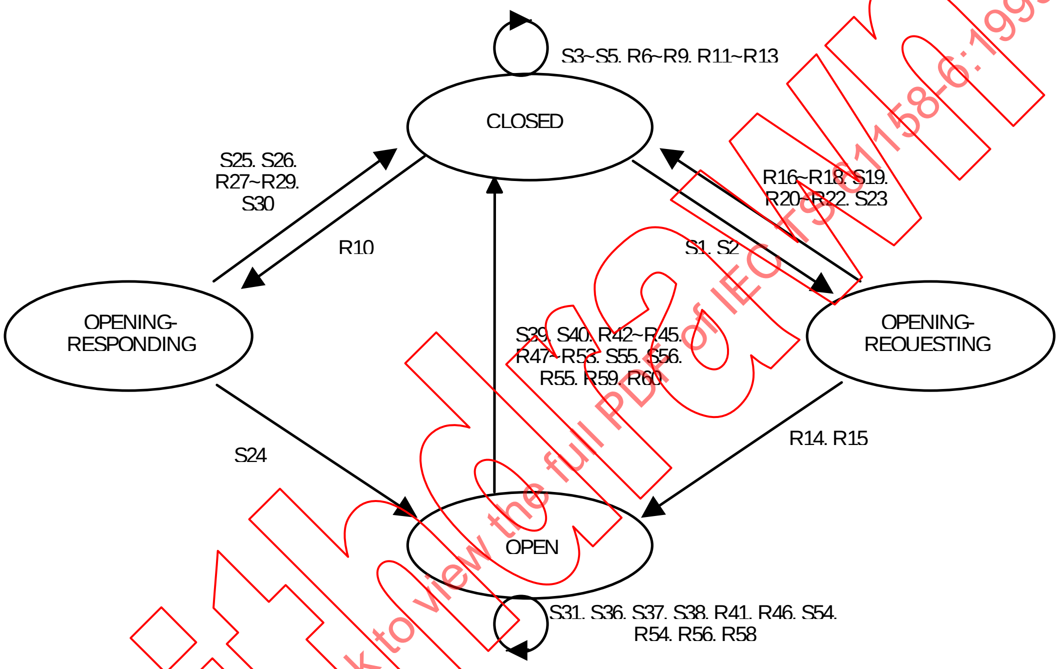

S3~S5. R6~R9. R11~R13
CLOSED
S25. S26. R27~R29. S30
R10
R16~R18. S19. R20~R22. S23
S1. S2
OPENING-RESPONDING
OPENING-REQUESTING
S39. S40. R42~R45. R47~R53. S55. S56. R55. R59. R60
S24
R14. R15
OPEN
S31. S36. S37. S38. R41. R46. S54. R54. R56. R58

Figure 2 – AP-AP Context Initiation State Machine

5.1.3 AP-AP Context Initiation State Transitions

In the tables of this subclause “Data” parameter for FAL Service Protocol Machine is expressed by “FalApduBody” to emphasize its semantics.

Table 4 – FAL State Machine Sender Transactions

#	Current State	Events Actions	Next State

S1	CLOSED	Initiate.req
		&& ApContextTest = “True”
		&& ApExplicitConnection = “True”
		
		EST.req {
		FalApduBody := “Initiate-RequestPDU”
		}
			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 (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 = “Initiate-ResponsePDU”
		}
			OPEN

#	Current State	Events Actions	Next State

S25	RSP	Initiate.rsp(-)
		
		EST.rsp(-) {
		FalApduBody = “Initiate-ErrorPDU”,
		ErrorCode = “ErrorCode of Initiate.rsp”
		},
		
		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 := “Confirmed-RequestPDU” }, OutstandingServicesRequestingCounter := OutstandingServicesRequestingCounter+1	OPEN

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

S37	OPEN	UnconfirmedService.req
		&& UnconfirmedServiceCheck = “True”
		&& PDU length ≤ NegotiatedMaxPduSizeSending
		&& RequestedServiceSupportedTest = “True”
		&& ImmediateAcknowledge = “True”
		
		UCA.req {
		FalApduBody := “Unconfirmed-PDU”
		}
			OPEN

#	Current State	Events Actions	Next State
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 section.	OPEN

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 := “Confirmed-ResponsePDU” }, OutstandingServicesRespondingCounter := OutstandingServicesRespondingCounter-1	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	Events Actions	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 5 – FAL State Machine Receiver Transactions

#	Current State	Events Actions	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 = “Initiate-RequestPDU”
		&& ApContextTest = “True”
		&& MaxFalPduLengthTest = "True"
		&& ServicesSupportedTest = “True”
		
		NegotiateOutstandingServices,
		
		Initiate.ind {
		}
			RSP

R11	CLOSED	EST.ind
		&& FalApduBody = “Initiate-RequestPDU”
		&& ApContextTest = “False”
		
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “AR error”
		}
			CLOSED

#	Current State	Events Actions	Next State

R12	CLOSED	EST.ind
		&& FalApduBody = "Initiate-RequestPDU"
		&& ApContextTest = "True"
		&& MaxFalPduLengthTest = "False"
		
		EST.rsp(-) {
		FalApduBody := "Initiate-ErrorPDU",
		ErrorCode := "Max-Fal-Fdu-Size-Insufficient"
		}
			CLOSED

R13	CLOSED	EST.ind
		&& FalApduBody = "Initiate-RequestPDU"
		&& ApContextTest = "True"
		&& MaxFalPduLengthTest = "True"
		&& ServicesSupportedTest = "False"
		
		EST.rsp(-) {
		FalApduBody := "Initiate-ErrorPDU",
		ErrorCode := "Service-Not-Supported"
		}
			CLOSED

R14	REQ	EST.cnf(+)
		&& FalApduBody = "Initiate-ResponsePDU"
		&& ApExplicitConnection = "True"
		
		Initiate.cnf(+) {
		}
			OPEN

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

R16	REQ	EST.cnf(-) && FalApduBody = "Initiate-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

#	Current State	Events Actions	Next State

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	Events Actions	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 = “Confirmed-ServicePDU” && PDU length ≤ NegotiatedMaxPduSizeReceiving && OutstandingServicesRespondingCounter < NegotiatedMaxOutstanding-ServicesResponding && InvokeIdExistent = “False” && IndicatedServiceSupportedTest = “True” ConfirmedService.ind { }, OutstandingServicesRespondingCounter := OutstandingServicesRespondingCounter+1	OPEN

R42	OPEN	CS.ind
		&& FalApduBody = “Confirmed-ServicePDU”
		&& PDU length > NegotiatedMaxPduSizeReceiving
		
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “PDU-size”
		},
		
		ResetArep
			CLOSED

#	Current State	Events Actions	Next State

R43	OPEN	CS.ind
		&& FalApduBody = "Confirmed-ServicePDU"
		&& 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 = "Confirmed-ServicePDU"
		&& 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 = "Confirmed-ServicePDU"
		&& 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 = "Unconfirmed-PDU"
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		&& IndicatedServiceSupportedTest = "True"
		
		UnconfirmedService.ind {
		}
			OPEN

#	Current State	Events Actions	Next State

R47	OPEN	UCS.ind
		&& FalApduBody = “Unconfirmed-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 = “Unconfirmed-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 = “Initiate-RequestPDU”
		
		Abort.req {
		Originator := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		Terminate.ind {
		LocallyGenerated := “True”,
		TerminateIdentifier := “FAL”,
		ReasonCode := “Connection-State-Conflict”
		},
		
		ResetArep
			CLOSED

#	Current State	Events Actions	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 = “Unconfirmed-PDU”
		&& PDU length ≤ NegotiatedMaxPduSizeReceiving
		
		UnconfirmedService.ind {
		}
			OPEN

#	Current State	Events Actions	Next State

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

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 section.	OPEN
R58	OPEN	CS.cnf && FalApduBody = “Confirmed-ResponsePDU” && PDU length ≤ NegotiatedMaxPduSizeReceiving && InvokeIdExistent = “True” && SameService = “True” ConfirmedService.cnf { }, OutstandingServicesRequestingCounter := OutstandingServicesRequestingCounter-1	OPEN

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

#	Current State	Events Actions	Next State

R60	OPEN	CS.cnf
		&& FalApduBody = “Confirmed-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

5.1.4 Functions

This subclause defines internal functions that are used by the AP AR Context state machine.

Table 6 – 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 7 – 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 8 – 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 9 – 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 10 – 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 11 – 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 12 – 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 13 – 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 14 – 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 AR-Compel — FCMP AR-Get Buffered Message — GBM AR-Status — FSTS AR-XON-OFF — AR_XON_OFF AR-RemoteRead — RR AR-RemoteWrite — RW		

Table 15 – 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 16 – 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 17 – 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 18 – 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 19 – 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 20 – 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 21 – 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.		

5.2 FAL Service Protocol Machine (FSPM)

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."

5.2.1 Primitive Definitions

5.2.1.1 Primitives Exchanged between AP_Context and FSPM

Table 22 – Primitives issued by AP_Context FAL User 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.
FCMP.req	AP_Context	Arep_Id	This primitive is used to convey an FAL-Compel (FCMP) request primitive from the AP_Context to the FSPM.
GBM.req	AP_Context	Arep_Id	This primitive is used to convey a Get-Buffered-Message (GBM) request primitive from the AP_Context to the FSPM.
RW.req	AP_Context	Arep_Id, Data, Priority	This primitive is used to remotely write the contents of a buffer.
RR.req	AP_Context	Arep_Id, Priority	This primitive is used to get the contents of a remote buffer.
UCA.req	AP_Context	Arep_Id, Remote_Dlsap_Address, Data	This primitive is used to send an unconfirmed service request.
AR_XON_OFF.req	AP_Context	Arep_Id, Remote_Dlsap_Address, Data	This primitive is used to perform flow control.

Table 23 – 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.
FCMP.cnf	FSPM	Arep_Id, Status	This primitive is used to convey an FAL-Compel (FCMP) confirmation primitive from the FSPM to the AP_Context.
GBM.cnf(+)	FSPM	Arep_Id, Duplicate_FAL-SDU, Data, Local_Timeliness, Remote_Timeliness	This primitive is used to convey a Get-Buffered-Message (GBM) positive confirmation primitive from the FSPM to the AP_Context.
GBM.cnf(-)	FSPM	Arep_Id	This primitive is used to convey a Get-Buffered-Message (GBM) negative confirmation primitive from the FSPM to the AP_Context.
FSTS.ind	FSPM	Arep_Id, Reported_Status	This primitive is used to convey an FAL-Status (FSTS) 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.
AR_XON_OFF.ind	FSPM	Arep_Id, Remote_DLSAP_Address, Data	This primitive is used to convey a flow control indication.
RW.cnf(+)	FSPM	Arep_Id, Status	This primitive is used to convey a Remote_Write (RW) positive confirmation primitive.
RW.cnf(-)	FSPM	Arep_Id, Status	This primitive is used to convey a Remote_Write (RW) negative confirmation primitive.
RR.cnf(+)	FSPM	Arep_Id, Data, Local_Timeliness, Remote_Timeliness	This primitive is used to convey a Remote_Read (RR) positive confirmation primitive.
RR.cnf(-)	FSPM	Arep_Id, Status	This primitive is used to convey a Remote_Read (RR) negative confirmation primitive.

5.2.1.2 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” clause.

5.2.2 FSPM State Tables

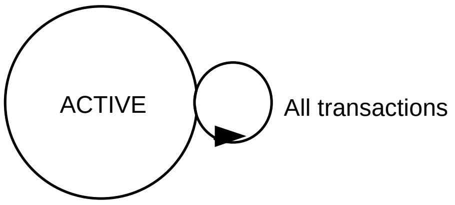


Figure 3 – State Transition Diagram of FSPM

Table 24 – FSPM State Table – Sender Transactions

#	Current State	Event 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

#	Current State	Event Action	Next State

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

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

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

S10	ACTIVE	RW.req
		&& SelectArep(Arep_Id) = ‘’True’’
		
		RW_req{
		user_data := Data,
		priority : = Priority
		}
			ACTIVE

S11	ACTIVE	RR.req
		&& SelectArep(Arep_Id) = ‘’True’’
		
		RR_req {
		priority : = Priority
		}
			ACTIVE

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

S13	ACTIVE	AR_XON_OFF.req
		&& SelectArep (Arep_Id) = "True"
		
		AR_XON_OFF_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 25 – FSPM State Table - Receiver Transactions

#	Current State	Event 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,
		    Duplicate_FAL-SDU := duplicate_fal_sdu,
		    Data := user_data,
		    Local_Timeliness := local_timeliness,
		    Remote_Timeliness := remote_timeliness
		  }
			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

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

#	Current State	Event Action	Next State

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

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

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

R12	ACTIVE	RW_cnf(-)
		
		RW.cnf(-) {
		Arep_Id := arep_id,
		Status := status
		}
			ACTIVE

R13	ACTIVE	RW_cnf(+)
		
		RW.cnf(+) {
		Arep_Id := arep_id,
		Status := status
		}
			ACTIVE

R14	ACTIVE	RR_cnf(-)
		
		RR.cnf(-) {
		Arep_Id := arep_id,
		Status := status
		}
			ACTIVE

R15	ACTIVE	RR_cnf(+)
		
		RR.cnf(+) {
		Arep_Id := arep_id,
		Data := user_data,
		Local_Timeliness := local_timeliness,
		Remote_Timeliness := remote_timeliness
		}
			ACTIVE

R16	ACTIVE	UCA_ind
		
		UCA.ind {
		Arep_Id := arep_id,
		Remote_DLSAP_Address := remote_dlsap_address,
		Duplicate_FAL-SDU := duplicate_fal_sdu,
		Data := user_data,
		}
			ACTIVE

#	Current State	Event Action	Next State

R17	ACTIVE	AR_XON_OFF_ind
		
		AR_XON_OFF.ind {
		Arep_Id := arep_id,
		Remote_DLSAP_Address := remote_dlsap_address,
		Data := user_data,
		}
			ACTIVE

5.2.2.1 Functions

Table 26 – 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.			

5.3 DLL Mapping Protocol Machine (DMPM)

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:

1. The ISA Data Link Layer specification 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.
2. "dl-identifiers" and "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:

1. 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 ISA Data Link Layer specification uses parameters to identify a queue or 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 specification does not include parameters that directly correspond to these buffer or queue identifiers. A means for identifying the buffers and queues between the FAL and the DL is a local matter.

A remark about initialization of the Data Link Layer:

The ISA Data Link Layer specification defines services to setup resources within the layer, such as DL-Create or DL-Bind services. Although they are useful to clarify the operations of the Data Link Layer, none of them affects protocol behavior of the FAL and DL. Therefore, the FAL assumes that such initialization procedures have been executed prior to the operations of the FAL state machines. For the same reason, these DL-services are not listed in annex C.

5.3.1 Primitive Definitions

5.3.1.1 Primitives Exchanged between DMPM and ARPM

Table 27 – 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, remote_dlsap_address, dll_priority, dls_user_data_timeliness, 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.

Table 28 – 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, calling_dlsap_address, dll_priority, fal_pdu, local_dle_timeliness, remote_dle_timeliness, sender_time_of_production	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, called_dlcep_address, called_address, calling_address, dlcep_class, delivery_from_responder, delivery_from_requestor, dll_priority, max_confirm_delay_on_connect, max_confirm_delay_on_data, dlpdu_authentication, residual_activity_as_sender, residual_activity_as_receiver, dlsdu_size_from_requestor, dlsdu_size_from_responder, sender_dl_timeliness_class, sender_time_of_production, receiver_dl_timeliness_class, dls_user_data	This primitive is used to convey a DL_Connect.ind primitive to the ARPM to process connection establishment. All the parameters that are associated with the DL_Connect.ind primitive are carried with this primitive.
Connect_cnf	DMPM	responding_address dlcep_class, delivery_from_responder, delivery_from_requestor, dll_priority, max_confirm_delay_on_connect, max_confirm_delay_on_data, dlpdu_authentication, residual_activity_as_sender, residual_activity_as_receiver, dlsdu_size_from_requestor, dlsdu_size_from_responder, sender_dl_timeliness_class, receiver_dl_timeliness_class, sender_time_of_production, 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 Data Link Layer to a designated ARPM.

5.3.1.2 Parameters of ARPM/DMPM Primitives

The parameters used with the primitives exchanged between the ARPM and the DMPM are described in table 29.

Table 29 – 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_dlcep_address	This parameter conveys the value of the RespondingAREP 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 Data Link Layer service name. Possible values are all the DL-XXXX.yyy primitives defined in this section and are represented as DMPM_XXXX_yyy.
dlcep_address	This parameter conveys the value of the dl_dlcep_address parameter.
duplicate_dlsdu	This parameter conveys the value of the dl_duplicate_dlsdu parameter of the received primitive.
dlcep_class	This parameter conveys the value of the dl_dlcep_class parameter.
dll_priority	This parameter conveys the value of the dl_dll_priority parameter.
delivery_from_responder	This parameter conveys the value of the dl_delivery_from_responder parameter.
delivery_from_requestor	This parameter conveys the value of the dl_delivery_from_requestor parameter.
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.
dlpdu_authentication	This parameter conveys the value of the dl_dlpdu_authentication parameter.
fal_pdu	This parameter conveys the value of the dl_dls_user_data parameter.
local_dlsap_address	This parameter conveys the value of the dl_local_dlsap_address parameter.
remote_dle_timeliness	This parameter conveys the value of the dl_sender_and_remote_dle_timeliness parameter.
local_dle_timeliness	This parameter conveys the value of the dl_dle_timeliness parameter.
max_confirm_delay_on_connect	This parameter conveys the value of the dl_max_confirm_delay_on_connect parameter.
max_confirm_delay_on_data	This parameter conveys the value of the dl_max_confirm_delay_on_data parameter.
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.
residual_activity_as_sender	This parameter conveys the value of the dl_residual_activity_as_sender parameter.
residual_activity_as_receiver	This parameter conveys the value of the dl_residual_activity_as_receiver parameter.
sender_dl_timeliness_class	This parameter conveys the value of the dl_sender_dl_timeliness_class parameter.
receiver_dl_timeliness_class	This parameter conveys the value of the dl_receiver_dl_timeliness_class parameter.
sender_time_of_production	This parameter conveys the value of the dl_time_of_production parameter.
remote_dlsap_address	This parameter conveys the value of the dl_remote_dlsap_address parameter.

5.3.1.3 Primitives Exchanged between Data Link Layer and DMPM

NOTE The following primitives and their parameters are defined in IEC 61158-3.

Table 30 – Primitives Exchanged between Data Link Layer and DMPM

Primitive Names	Source	Associated Parameters and Their Types
DL_Connect.ind	Data Link Layer	dl_called_address, dl_calling_address, dl_dlcep_class, dl_delivery_from_requestor, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlpdu_authentication, dl_residual_activity_as_sender, dl_residual_activity_as_receiver, dl_dlsdu_size_from_requestor, dl_dlsdu_size_from_responder, dl_sender_dl_timeliness_class, dl_sender_time_of_production, dl_receiver_dl_timeliness_class, dl_dls_user_data
DL_Connect.req(out)	Data Link Layer	(none)
DL_Connect.cnf		dl_responding_address, dl_dlcep_class, dl_delivery_from_requestor, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlpdu_authentication, dl_residual_activity_as_sender, dl_residual_activity_as_receiver, dl_dlsdu_size_from_requestor, dl_dlsdu_size_from_responder, dl_sender_dl_timeliness_class, dl_receiver_dl_timeliness_class, dl_sender_dl_timeliness_class, dl_sender_time_of_production, dl_receiver_dl_timeliness_class, dl_dls_user_data
DL_Connection_Established.ind	Data Link Layer	(none)
DL_Disconnect.ind	Data Link Layer	dl_originator, dl_reason, dl_dls_user_data
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_duplicate_dlsdu
DL_Unitdata.ind	Data Link Layer	dl_called_address, dl_calling _address, dl_dll_priority, dl_dls_user_data
DL_Unitdata.cnf	Data Link Layer	dl_status
DL_Buffer_Sent.ind	Data Link Layer	(none)
DL_Get.req(out)	Data Link Layer	dl_status, dl_reported_service_identification_class, dl_calling_dlsap_address, dl_dll_priority, dl_dls_user_data, dl_local_dle_timeliness, dl_sender_and_remote_dle_timeliness, dl_sender_time_of_production
DL_Put.req(out)	Data Link Layer	dl_status
DL_Compel_Service.req(out)	Data Link Layer	dl_status

DL_Connect.req(in)	DMPM	dl_called_address, dl_calling_address, dl_dlcep_address, dl_dlcep_class, dl_delivery_from_requestor, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlpdu_authentication, dl_residual_activity_as_sender, dl_residual_activity_as_receiver, dl_scheduling_policy, dl_dlsdu_size_from_requestor, dl_dlsdu_size_from_responder, dl_sender_dl_timeliness_class, dl_sender_time_window_size, dl_sender_synchronizing_dlcep, dl_sender_time_of_production, dl_receiver_dl_timeliness_class, dl_receiver_time_window_size, dl_receiver_synchronizing_dlcep, dl_dls_user_data
DL_Connect.rsp	DMPM	dl_responding_address, dl_dlcep_address, dl_dlcep_class, dl_delivery_from_requestor, dl_delivery_from_responder, dl_dll_priority, dl_max_confirm_delay_on_connect, dl_max_confirm_delay_on_data, dl_dlpdu_authentication, dl_residual_activity_as_sender, dl_residual_activity_as_receiver, dl_scheduling_policy, dl_dlsdu_size_from_requestor, dl_dlsdu_size_from_responder, dl_sender_dl_timeliness_class, dl_sender_time_window_size, dl_sender_synchronizing_dlcep, dl_sender_time_of_production , dl_receiver_dl_timeliness_class, dl_receiver_time_window_size, dl_receiver_synchronizing_dlcep, dl_dls_user_data
DL_Unitdata.req	DMPM	dl_called_address, dl_calling _address, dl_dll_priority, dl_max_confirm_delay, dl_remote_dle_confirmed, dl_dls_user_data
DL_Disconnect.req	DMPM	dl_reason dl_dls_user_data
DL_Data.req	DMPM	dl_dls_user_data
DL_Get.req(in)	DMPM	(none)
DL_Put.req(in)	DMPM	dl_dls_user_data, dl_dls_user_data_timeliness
DL_Compel_Service.req(in)	DMPM	dl_action_class, dl_remote_dlsap_address, dl_priority

5.3.1.4 Parameters of DMPM/Data Link Layer Primitives

The parameters used with the primitives exchanged between the DMPM and the Data Link Layer are identical to those defined in the Data Link Layer Service Definition (IEC 61158-3). They are prefixed by “dl_” to indicate that they are used by the FAL.

5.3.2 DMPM State Machine

5.3.2.1 DMPM States

The defined state of the DMPM together with its description are listed in table 31.

Table 31 – 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.

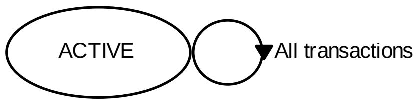


Figure 4 – State Transition Diagram of DMPM

5.3.2.2 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 32 – DMPM State Table - Sender Transactions

#	Current State	Event Action	Next State

S1	ACTIVE	FAL-PDU_req
		&& dmpm_service_name = "DMPM_Connect_req"
		
		PickArep (arep_id),
		
		DL_Connect.req(in) {
		dl_called_address := called_address,
		dl_calling_address := calling_address,
		dl_dlcep_address := local_dlcep_address,
		dl_dlcep_class := DlcepClass,
		dl_delivery_from_requestor := FromRequestorToResponder,
		dl_delivery_from_responder := FromResponderToRequestor,
		dl_dll_priority := DllPriority,
		dl_max_confirm_delay_on_connect := MaxConfirmDelayOnDlConnect,
		dl_max_confirm_delay_on_data := MaxConfirmDelayOnDlData,
		dl_dlpdu_authentication := DlpduAuthentication,
		dl_residual_activity_as_sender := ResidualActivityAsSender,
		dl_residual_activity_as_receiver := ResidualActivityAsReceiver,
		dl_scheduling_policy := DlSchedulingPolicy,
		dl_dlsdu_size_from_requestor := MaxDlsduSizeFromRequestor,
		dl_dlsdu_size_from_responder := MaxDlsduSizeFromResponder,
		
		dl_sender_dl_timeliness_class := PublisherDlTimelinessClass,
		dl_sender_time_window_size := PublisherTimeWindowSize,
		dl_sender_synchronizing_dlcep := PublisherSynchronizingDlcep,
		dl_time_of_production := TimeOfProduction,
		dl_receiver_dl_timeliness_class := SubscriberDlTimelinessClass,
		dl_receiver_time_window_size := SubscriberTimeWindowSize,
		dl_receiver_synchronizing_dlcep := SubscriberSynchronizingDlcep,
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S2	ACTIVE	FAL-PDU_req
		&& dmpm_service_name = "DMPM_Connect_rsp"
		
		PickArep (arep_id),
		
		DL_Connect.rsp {
		dl_responding_address := responding_address,
		dl_dlcep_address := local_dlcep_address,
		dl_dlcep_class := DlcepClass,
		dl_delivery_from_requestor := FromRequestorToResponder,
		dl_delivery_from_responder := FromResponderToRequestor,
		dl_dll_priority := DllPriority,
		dl_max_confirm_delay_on_connect := MaxConfirmDelayOnDlConnect,
		dl_max_confirm_delay_on_data := MaxConfirmDelayOnDlData,
		dl_dlpdu_authentication := DlpduAuthentication,
		dl_residual_activity_as_sender := ResidualActivityAsSender,
		dl_residual_activity_as_receiver := ResidualActivityAsReceiver,
		dl_scheduling_policy := DlSchedulingPolicy,
		dl_dlsdu_size_from_requestor := MaxDlsduSizeFromRequestorNegotiated,
		dl_dlsdu_size_from_responder := MaxDlsduSizeFromResponderNegotiated,
		dl_sender_dl_timeliness_class := PublisherDlTimelinessClass,
		dl_sender_time_window_size := PublisherTimeWindowSize,
		dl_sender_synchronizing_dlcep := PublisherSynchronizingDlcep,
		dl_sender_time_of_production := TimeOfProduction,
		dl_receiver_dl_timeliness_class := SubscriberDlTimelinessClass,
		dl_receiver_time_window_size := SubscriberTimeWindowSize,
		dl_receiver_synchronizing_dlcep := SubscriberSynchronizingDlcep,
		dl_dls_user_data := dlsdu
		}
			ACTIVE

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

S4	ACTIVE	FAL-PDU_req
		&& dmpm_service_name = "DMPM_Data_req"
		
		PickArep (arep_id),
		
		DL_Data.req {
		dl_dls_user_data := dlsdu
		}
			ACTIVE

S5	ACTIVE	FAL-PDU_req
		&& dmpm_service_name = "DMPM_Unitdata_req"
		
		PickArep (arep_id),
		
		DL_Unitdata.req {
		dl_called_address := RemoteDlsapAddress,
		dl_calling_address := LocalDlsapAddress,
		dl_dll_priority := DllPriority,
		dl_max_confirm_delay := MaxConfirmDelayOnUnitdata,
		dl_remote_dle_confirmed := DleRemoteConf, *1
		dl_dls_user_data := dlsdu
		}
		
		*1: Default “False”
			ACTIVE

S6	ACTIVE	FAL-PDU_req
		&& dmpm_service_name = "DMPM_Put_req"
		
		PickArep (arep_id),
		
		DL_Put.req(in) {
		dl_dls_user_data := dlsdu,
		dl_dls_user_data_timeliness := dls_user_data_timeliness
		}
		
		DL_Put.req(out)
		&& dl_status = “success”
		FAL-PDU_ind {
		dmpm_service_name := “DMPM_Put_cnf”,
		reason := dl_status
		}
		
		DL_Put.req(out)
		&& dl_status <> “success”
		FAL-PDU_ind {
		dmpm_service_name := “DMPM_Put_cnf”,
		reason := dl_status
		}
		
		ErrorToARPM {
		originator := “local_dls”,
		reason := dl_status
		}
			ACTIVE

S7	ACTIVE	FAL-PDU_req
		&& dmpm_service_name = “DMPM_Get_req”
		
		    PickArep (arep_id),
		
		    DL_Get.req(in) {
		    }
		
		DL_Get.req(out)
		&& dl_reported_service_identification_class = "NONE"
		
		    FAL-PDU_ind {
		        dmpm_service_name := "DMPM_Get_cnf",
		        reason := dl_status,
		        calling_dlsap_address := dl_calling_dlsap_address,
		        fal_pdu := dl_,
		        local_dle_timeliness := dl_local_dle_timeliness,
		        remote_dle_timeliness := dl_sender_and_remote_dle_timeliness,
		        sender_time_of_production := dl_sender_time_of_production
		    }
			ACTIVE

S8	ACTIVE	FAL-PDU_req
		&& dmpm_service_name = “DMPM_Compel_req”
		
		    PickArep (arep_id),
		
		    DL_Compel_Service.req(in) {
		        dl_action_class := action_class
		        dl_remote_dlsap_address := remote_dlsap_address,
		        dl_priority := dll_priority
		    }
		
		DL_Compel_Service.req(out)
		
		    FAL-PDU_ind {
		        dmpm_service_name := "DMPM_Compel_Service_cnf",
		        reason := dl_status
		    }
			ACTIVE

Table 33 – DMPM State Table - Receiver Transactions

#	Current State	Event Action	Next State

R1	ACTIVE	DL_Connect.ind
		&& LocateArep (dl_dls_user_data) = “NOT FOUND”
		
		    DL_Disconnect.req {
		        dl_reason := “AREP Not Found”,
		        dl_dls_user_data := “null”
		    }
			ACTIVE

R2	ACTIVE	DL_Connect.ind
		&& LocateArep (dl_dls_user_data) = “NOT Relevant”
		
		    DL_Disconnect.req {
		        dl_reason := “Invalid AREP Type”,
		        dl_dls_user_data := “null”
		    }
			ACTIVE

R3	ACTIVE	DL_Connect.ind
		&& LocateArep (dl_dls_user_data) = “True”
		
		Connect_ind {
		calling_dlcep_address := RequestingAREP, ; from dlsdu of DL_Connect.ind
		called_dlcep_address := RespondingAREP, ; from dlsdu of DL_Connect.ind
		called_address := dl_called_address,
		calling_address := dl_calling_address,
		dlcep_class := dl_dlcep_class,
		delivery_from_requestor := dl_delivery_from_requestor,
		delivery_from_responder := dl_delivery_from_responder,
		dll_priority := dl_dll_priority,
		max_confirm_delay_on_connect := dl_max_confirm_delay_on_connect,
		max_confirm_delay_on_data := dl_max_confirm_delay_on_data,
		dlpdu_authentication := dl_dlpdu_authentication,
		residual_activity_as_sender := dl_residual_activity_as_sender,
		residual_activity_as_receiver := dl_residual_activity_as_receiver,
		dlsdu_size_from_requestor := dl_dlsdu_size_from_requestor,
		dlsdu_size_from_responder := dl_dlsdu_size_from_responder,
		sender_dl_timeliness_class := dl_sender_dl_timeliness_class,
		sender_time_of_production := dl_sender_time_of_production,
		receiver_dl_timeliness_class := dl_receiver_dl_timeliness_class,
		dls_user_data := dl_dls_user_data
		}
			ACTIVE

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

R5	ACTIVE	DL_Connect.cnf
		&& FindAREP () = "True"
		
		Connect_cnf {
		responding_address := dl_responding_address,
		dlcep_class := dl_dlcep_class,
		delivery_from_requestor := dl_delivery_from_requestor,
		delivery_from_responder := dl_delivery_from_responder,
		dll_priority := dl_dll_priority,
		max_confirm_delay_on_connect := dl_max_confirm_delay_on_connect,
		max_confirm_delay_on_data := dl_max_confirm_delay_on_data,
		dlpdu_authentication := dl_dlpdu_authentication,
		residual_activity_as_sender := dl_residual_activity_as_sender,
		residual_activity_as_receiver := dl_residual_activity_as_receiver,
		dlsdu_size_from_requestor := dl_dlsdu_size_from_requestor,
		dlsdu_size_from_responder := dl_dlsdu_size_from_responder,
		sender_dl_timeliness_class := dl_sender_dl_timeliness_class,
		sender_time_of_production := dl_sender_time_of_production,
		receiver_dl_timeliness_class := dl_receiver_dl_timeliness_class,
		dls_user_data := dl_dls_user_data
		}
			ACTIVE

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

R7	ACTIVE	DL_Connection_Established.ind
		&& FindAREP () = "True"
		
		FAL-PDU_ind {
		dmpm_service_name := "DMPM_Connection_Established_ind",
		fal_pdu := "null"
		}
			ACTIVE

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

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

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

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

R12	ACTIVE	DL_Unitdata.ind
		&& FindAREP (dl_called_address) = "True"
		&& ExplicitQueue = "True"
		
		    DL_Get.req(in) {
		    }
		
		DL_Get.req(out)
		&& dl_status = "success"
		&& dl_reported_service_identification_class = "DL(SAP)-ADDRESS"
		
		    FAL-PDU_ind {
		        dmpm_service_name := "DMPM_Unitdata_ind",
		        calling_address := dl_calling_dlsap_address,
		        dll_priority := dl_dll_priority,
		        fal_pdu := dl_dls_user_data,
		        local_dle_timeliness := dl_local_dle_timeliness,
		        remote_dle_timeliness := dl_sender_and_remote_dle_timeliness,
		        sender_time_of_production := dl_sender_time_of_production
		    }
		
		DL_Get.req(out)
		&& dl_status <> "success"
		
		    FAL-PDU_ind {
		        dmpm_service_name := “DMPM_Unitdata_ind”,
		        reason := dl_status
		    }
		
		    ErrorToARPM {
		        originator := “local_dls”,
		        reason := dl_status
		    }
			ACTIVE

R13	ACTIVE	DL_Unitdata.cnf
		&& dl_status <> "success"
		
		    FAL-PDU_ind {
		        dmpm_service_name := “DMPM_Unitdata_cnf”,
		        reason := dl_status
		    }
		    ErrorToARPM {
		        originator := “local_dls”,
		        reason := dl_status
		    }
			ACTIVE

R14	ACTIVE	DL_Unitdata.cnf
		&& dl_status = "success"
		
		FAL-PDU ind {
		        dmpm service name := "DMPM_Unidata_cnf",
		        reason := dl_status
		}
			ACTIVE

R15	ACTIVE	DL_Data.ind && FindAREP () = "False" (no actions taken)	ACTIVE

R16	ACTIVE	DL_Data.ind
		&& FindAREP () = "True"
		&& ExplicitQueue = "False"
		
		FAL-PDU_ind {
		dmpm_service_name := "DMPM_Data_ind",
		fal_pdu := dl_dls_user_data
		}
			ACTIVE

R17	ACTIVE	DL_Data.ind
		&& FindAREP () = "True"
		&& ExplicitQueue = "True"
		
		DL_Get.req(in) {
		}
		
		DL_Get.req(out)
		&& dl_status = "success"
		&& dl_reported_service_identification_class = "DLCEP"
		
		FAL-PDU_ind {
		dmpm_service_name := "DMPM_Data_ind",
		fal_pdu := dl_dls_user_data
		}
		
		DL_Get.req(out)
		&& dl_status <> "success"
		
		FAL-PDU_ind {
		dmpm_service_name := “DMPM_Get_cnf”,
		reason := dl_status
		}
		
		ErrorToARPM {
		originator := “local_dls”,
		reason := dl_status
		}
			ACTIVE

R18	ACTIVE	DL_Data.cnf && dl_status = "success" (no actions taken)	ACTIVE

R19	ACTIVE	DL_Data.cnf
		&& dl_status <> "success"
		
		FAL-PDU_ind {
		dmpm_service_name := “DMPM_Data_cnf”,
		reason := dl_status
		}
		ErrorToARPM {
		originator := “local_dls”,
		reason := dl_status
		}
			ACTIVE

R20	ACTIVE	DL_Buffer_Received.ind && FindAREP () = "False" (no actions taken)	ACTIVE

R21	ACTIVE	DL_Buffer_Received.ind
		&& FindAREP () = "True"
		
		  DL_Get.req(in) {
		  }
		
		DL_Get.req(out)
		&& dl_status = "success"
		&& dl_reported_service_identification_class = "NONE"
		
		  FAL-PDU_ind {
		    dmpm_service_name := "DMPM_Buffer_Received_ind",
		    duplicate_dlsdu :=dl_duplicate_dlsdu, ; from DL_Buffer_Received.ind
		    fal_pdu := dl_dls_user_data,
		    local_dle_timeliness := dl_local_dle_timeliness,
		    remote_dle_timeliness := dl_sender_and_remote_dle_timeliness,
		    sender_time_of_production := dl_sender_time_of_production
		  }
		
		DL_Get.req(out)
		&& dl_status <> "success"
		
		  FAL-PDU_ind {
		    dmpm_service_name := “DMPM_Get_cnf”,
		    reason := dl_status
		  }
		
		  ErrorToARPM {
		    originator := “local_dls”,
		    reason := dl_status
		  }
			ACTIVE

R22	ACTIVE	DL_Buffer_Sent.ind && FindAREP () = "False" (no actions taken)	ACTIVE

R23	ACTIVE	DL_Buffer_Sent.ind
		&& FindAREP () = "True"
		
		  FAL-PDU_ind {
		    dmpm_service_name := “DMPM_Buffer_Sent_ind”
		  }
			ACTIVE

5.3.2.3 Functions used by DMPM

Table 34 – Function PickArep

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

Table 35 – Function FindAREP

Name	FindAREP	Used in	DMPM
Input		Output	
dl_called_address || (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 36 – Function LocateArep

Name	LocateArep	Used in	DMPM
Input		Output	
dl_dls_user_data (of DL_Connect.ind)		True || Not Found || Not Relevant Access to attributes of the located AREP.	
Function			
This function returns the value of True and a means to get access to the attributes of the located AREP if all of the following conditions are met. Otherwise, it returns the value of either “Not Found” or “Not Relevant.” a) Decodes the dlsdu that is conveyed by the dl_dls_user_data argument and checks if the FAL PDU type is “EST_ReqPDU.” b) Decodes the FAL-PDU and extracts the RequestingAREP and RespondingAREP parameters. c) Looks for the AREPs that can accept a peer-to-peer connection, such as a QUB AREP, whose LocalDlcepAddress attribute value is equal to the RespondingAREP.			

Annex A
(normative)

FAL Header

A.1 Introduction

All the FAL PDUs shall have the common PDU-header called FalArHeader. The FalArHeader identifies abstract syntax, transfer syntax, and each of the PDUs. This normative annex defines how this header shall be used.

NOTE The structure of the FalArHeader is defined in the main text of this technical specification.

FAL Protocol Specifier (Bit 8) is always zero (0) for the protocol defined by this technical specification.

Table A.1 – FAL Header

Bit position of the FalArHeader			**Abstract Syntax**	**Encoding Rule**	**PDU Type**	**Revision**
8	7 6 5 4 3	2 1				
0	0 0 0 0 0	0 0	FAL AS2	TER	Establish-RequestPDU	Revision1
0	0 0 0 0 0	0 1	FAL AS2	TER	Establish-ResponsePDU	Revision1
0	0 0 0 0 0	1 0	FAL AS2	TER	Establish-ErrorPDU	Revision1
0	0 0 0 0 0	1 1	FAL AS2	TER	AbortPDU	Revision1
0	0 0 0 0 1	0 0	FAL AS2	TER	ConfirmedSend-RequestPDU	Revision1
0	0 0 0 0 1	0 1	FAL AS2	TER	ConfirmedSend-ResponsePDU	Revision1
0	0 0 0 0 1	1 0	FAL AS2	TER	UnconfirmedSendPDU	Revision1
0	1 0 0 0 0	0 0	FAL AS1	CER	Establish-CommandPDU	Revision1
0	1 0 0 0 0	0 1	FAL AS1	CER	Establish-AffirmativePDU	Revision1
0	1 0 0 0 0	1 0	FAL AS1	CER	Establish-NegativePDU	Revision1
0	1 0 0 0 0	1 1	FAL AS1	CER	AbortPDU	Revision1
0	1 0 0 0 1	0 0	FAL AS1	CER	ConfirmedSend-CommandPDU	Revision1
0	1 0 0 0 1	0 1	FAL AS1	CER	ConfirmedSend-AffirmativePDU	Revision1
0	1 0 0 0 1	1 0	FAL AS1	CER	ConfirmedSend-NegativePDU	Revision1
0	1 0 0 0 1	1 1	FAL AS1	CER	UnconfirmedSend-CommandPDU	Revision1
0	1 0 0 1 0	0 0	FAL AS1	CER	UnconfirmedAcknowledgedSend-CommandPDU	Revision1
0	1 0 0 1 0	0 1	FAL AS1	CER	UnconfirmedAcknowledgedSend-AffirmativePDU	Revision1
0	1 0 0 1 0	1 0	FAL AS1	CER	IdleSend-CommandPDU	Revision1
0	1 0 0 1 0	1 1	FAL AS1	CER	AR-XON-OFF-CommandPDU	Revision1
0	0 1 0 S 0	0 0	FAL AS1	MER	Establish-CommandPDU	Revision1
0	0 1 0 S 0	0 1	FAL AS1	MER	Establish-AffirmativePDU	Revision1
0	0 1 0 S 0	1 0	FAL AS1	MER	Establish-NegativePDU	Revision1
0	0 1 0 S 0	1 1	FAL AS1	MER	AbortPDU	Revision1
0	0 1 0 S 1	0 0	FAL AS1	MER	ConfirmedSend-CommandPDU	Revision1
0	0 1 0 S 1	0 1	FAL AS1	MER	ConfirmedSend-AffirmativePDU	Revision1
0	0 1 0 S 1	1 0	FAL AS1	MER	ConfirmedSend-NegativePDU	Revision1
0	0 1 0 S 1	1 1	FAL AS1	MER	UnconfirmedSend-CommandPDU	Revision1
0	1 1 0 S 0	0 0	FAL AS1	BER	BufferedDataTransferPDU	Revision1
0	0 0 1 0 0	0 0	FAL AS2	TER	UnconfirmedAcknowledgedSend-CommandPDU	Revision1
0	0 0 1 0 0	0 1	FAL AS2	TER	UnconfirmedAcknowledgedSend-AffirmativePDU	Revision1
0	0 0 1 0 0	1 0	FAL AS2	TER	IdleSend-CommandPDU	Revision1
0	0 0 1 0 0	1 1	FAL AS2	TER	AR-XON-OFF-CommandPDU	Revision1
0	0 0 1 0 1	0 0	FAL AS2	TER	Establish-Request2PDU	Revision1
1	X X X X X	X X			Reserved	

-- All other code points are reserved for additional protocols and future revisions.

-- S-bit (bit4) is used for segmentation purposes only. It indicates to the receiver of an FAL PDU, if this PDU is the last segment (S = 0) or not (S = 1).

Annex B
(normative)

Reason Codes

B.1 Introduction

This annex specifies the values of the Reason parameter of the Data Link Layer DL-Disconnect service that are supplied by the FAL.

Table B.1 – Reason Codes

FAL Reason parameter	Value
Invalid FAL-PDU	31 (Hex)
Remote Address Mismatch	32 (Hex)
Multiple Initiators	33 (Hex)
Invalid DL-Event	34 (Hex)
AREP Busy	35 (Hex)
AREP Not Found	36 (Hex)
Invalid AREP Type	37 (Hex)
DLCEP Not Found	38 (Hex)
Invalid DI PDU	39 (Hex)
Not allowed AREP primitive in connection establishment	3A (Hex)
Number of parallel services exceeded	3B (Hex)
Not allowed service as server	3C (Hex)
Not allowed service as client	3D (Hex)
Context Check Negative	3E (Hex)
Invalid FAL-PDU	3F (Hex)
Invalid Event for Role	40 (Hex)
UCA_AckPDU received and UCC=0	41 (Hex)
RCTimer expired	42 (Hex)
RSTimer expired	43 (Hex)
MCTimer expired	44 (Hex)
MSTimer expired	45 (Hex)
Execution error in cyclic data transfer	46 (Hex)

Annex C
(normative)

Data Link Layer Service Selection

C.1 Introduction

This annex briefly describes the Data Link Layer services utilized by the FAL. These Data Link Layer services are fully defined in the Data Link Layer specification (IEC 61158-3).

NOTE The FAL assumes that resources, such as buffers or queues, are set up prior to any operations of FAL protocol machines by a local means. DL services such as Create or Bind may be used for this purpose, the use of these services is not required by the FAL. Therefore, these local services are not listed in this annex.

C.1.1 DL-Connect

This service is used to establish a new connection or join in an existing one, and refers to the Connect service.

C.1.2 DL-Connection-Established

The DL-Connection-Established service is used to inform the FAL which has sent an ASC.rsp(+) primitive that a Data Link Layer connection is ready to use.

C.1.3 DL-Disconnect

This service is used to release an existing connection or leave from it and refers to the Disconnect service.

C.1.4 DL-Unitdata

This service is used for the connectionless data transfer mode and refers to the Unitdata service. It is used to transmit an FAL-PDU from one AREP.

C.1.5 DL-Data

This service is used for connection-oriented data transfer mode and refers to the Data service. This service is used to transfer an FAL-PDU from one AREP.

C.1.6 DL-Put

This service is used to copy an FAL-PDU to a buffer. It refers to the Put service.

C.1.7 DL-Get

This service is used to read an FAL-PDU from a buffer, or to attempt to remove an FAL-PDU from a FIFO queue, which has been bound to a DLSAP. It refers to the Get service.

C.1.8 DL-Buffer-Received

The DL-Buffer-Received service is used to inform the FAL of a new update on the specified receive buffer.

C.1.9 DL-Buffer-Sent

The DL-Buffer-Sent service is used to inform the FAL that the specified buffer content has just been sent.

C.1.10 DL-Compel-Data

The DL-Compel-Data service is used to compel transmission of the content of the buffer of the specified DLCEP.

Annex D
(normative)

Compact Encoding Rule (CER)

D.1 Compact Encoding Rule (CER)

D.1.1 Introduction

The APDUs that conform to this technical specification shall be encoded in a uniform format as shown in the figure below. The APDUs consist of two major parts: the "APDU Header" part and the "APDU Body" part.

(1)	(1 or two)	(1)	(n) --- Octets
FalArHeader Field	Type Field	(InvokeID)*	Service Specific Parameters
<------------------- APDU Header	---------------->		<-------------------- APDU Body --------------------->

NOTE The presence of the InvokeID Field depends on the APDU type.

Figure D.1 – APDU Overview

In order to realize an efficient APDU while maintaining flexible encoding, different encoding rules are used for the APDU Header part and the APDU Body part.

NOTE The ISA Data Link layer service provides a DLSDU parameter which implies the length of the APDU. Hence, the APDU length information is not included in the APDU.

D.1.2 APDU Header Encoding

The APDU Header part is always present in all APDUs which conform to this technical specification. It consists of three fields: the AR Control Field, the Type Field, and the optional InvokeID Field. They are shown in figure D.1.

D.1.2.1 APDU Header Encoding

Refer to annex A for the encoding rule of the FalArHeader field.

D.1.2.2 Encoding of Type Field

1) Both the APDU type and the service type are encoded in the Type Field that is always the second of the APDUs.

2) All bits of the Type Field are used to encode the service type.

2.1) The service types are encoded in bits 8 to 1 of the Type Field, with bit 8 the most significant bit and bit 1 the least significant bit. The range of the service type shall be between 0 (zero) and 254, inclusive.

2.2) The value of 255 is reserved for future extensions to this standard.

2.3) The service types are specified in the abstract syntax as a positive integer value. This positive integer value shall be encoded in the bits described in 2.1), where the most significant bit of a positive integer value shall be aligned to bit 8 and its least significant bit to bit 1.

3) Figure D.2 illustrates the encoding of the Type Field.

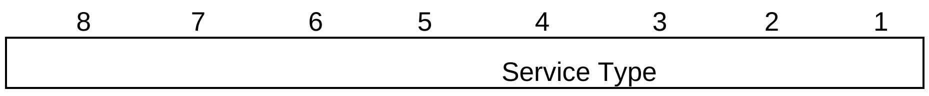


Figure D.2 – Type Field

D.1.2.3 Encoding of InvokeID Field

1) The InvokeID Field shall be present if it is indicated in the abstract syntax. Otherwise, this field shall not be present. If present, the Invoke ID parameter supplied by a service primitive shall be placed in this field.

D.1.2.4 APDU Body Encoding

The FAL encoding rules are based on terms and conventions defined in ISO/IEC 8825. The encoding consists of three components in the following order:

- An Identifier Octet
- Length Octet(s)
- The Contents Octets

The Identifier Octet or Length Octet(s), or both, may be removed from the encoding, depending data type of the field of the APDUs being encoded. The rules for removal are defined later.

D.1.2.4.1 Identifier Octet

1) The Identifier Octet shall encode the tag defined in the FAL Abstract Syntax and shall consist of one octet.

1.1) The Identifier Octet comprises a class and a number. The class encodes the class of the FAL tag, whereas a number encodes a number associated to each tag.

2) Two classes are defined for the Identifier Octet: The Context-Specific class and the FAL-Specific class.

2.1) The scope of a Context-Specific class is limited to the construction in which it is used. The same tag may be used to represent different types in different productions.

2.2) The scope of an FAL-Specific identifier is global to the module definitions of the Abstract Syntax definition. The same tag shall designate the same type within the same module.

2.3) Bit 8 of the Identifier Octets shall indicate its class. If it is set to zero (0), the class shall be Context-Specific. If it is set to one (1), the class shall be FAL-Specific.

3) The bits 7 to 1 of the Identifier Octet shall represent the value of the tag, with bit 7 being the most significant bit and bit 1 the least significant bit. This gives tags with the possible values of 0 (zero) through 126 inclusive, for each of the classes.

4) The value of 127 is reserved for future extensions this technical specification.

5) The convention for expressing an FAL-specific identifier with a tagged value of two shall be FAL-Specific 2.

6) When necessary, the abstract syntax shall explicitly specify which class shall be used. The default class shall be "Context Specific."

7) Figures D.3 and D.4 depict encoding examples of Identifier Octets.

8	7	6	5	4	3	2	1
0	Tag Number						

Figure D.3 – Identifier Octet (Context-Specific)

8	7	6	5	4	3	2	1
1	Tag Number						

Figure D.4 – Identifier Octet (FAL-Specific)

D.1.2.4.2 Length Octet(s)

1) The Length Octet(s) shall consist of one octet or three octets.

1.1) If the value of the first Length Octet is other than 255, there shall be no subsequent Length Octets and the first octet shall contain the value for the Length Octet defined later.

1.2) If the value of the first Length Octet is 255, there shall be two subsequent Length Octets that shall contain the value for the Length Octets defined later. In this case, the length information of the Contents Octets shall be represented by the last two octets of the Length Octets, where the most significant bit of the second of three Length Octet(s) shall be the most significant bit of the length value and the least significant bit of the third of the three Length Octet(s) shall be the least significant bit of the length value.

2) It shall be the sender's option to use either of the Length Octet(s) formats. The three octets format may be used to convey a length value of one, for instance.

3) The meaning of the Length Octet(s) depends on the type of the value being encoded. If the encoding of the Contents Octets is primitive, the Length Octet(s) shall contain the number of octets in the Contents Octets. If the encoding of the Contents Octets is constructed, the Length Octet(s) shall contain the number of the first level components of the Contents Octets.

4) Figures D.5 and D.6 depict encoding examples of the Length Octet(s):

8	7	6	5	4	3	2	1
(msb)	value of the Length Octet as defined in (3) above						(lsb)

Figure D.5 – Length Octet (one-octet format)

first octet	15	second and third octets	1
1 1 1 1 1 1 1 1	(msb)	value of the Length Octets as defined in (3) above	(lsb)

Figure D.6 – Length Octet (three-octet format)

D.1.2.4.3 Contents Octets

1) The Contents Octets shall encode the data value according to the encoding rule defined for its type.

2) The Contents Octets shall have either of the following two forms: a primitive encoding or a constructed encoding.

2.1) If the Contents Octets contain a primitive encoding, they represent an encoding of one value.

2.2) If the Contents Octets contain a constructed encoding, they represent an enumerated encoding of more than one value.

D.1.3 Data Type Encoding Rules (Base Encoding)

The base encoding rules of the following primitive data types defined in IEC 61158-5 are specified in this subclause:

- Boolean
- NULL
- Integer, Integer8, Integer16, Integer32
- Unsigned, Unsigned8, Unsigned16, Unsigned32
- Floating32, Floating64
- BitString, BitString8, BitString16, BitString32
- OctetString, OctetString2, OctetString4, OctetString8, OctetString16
- VisibleString, VisibleString2, VisibleString4, VisibleString8, VisibleString16
- ISO10646String
- UniversalTime
- BinaryTime0, BinaryTime1, BinaryTime2, BinaryTime3, BinaryTime4
- BinaryTime5, BinaryTime6, BinaryTime7, BinaryTime8, BinaryTime9
- CompactBCDArray
- BCD
- FieldbusTime
- CompactBooleanArray
- ANY

The base encoding rules of the following constructed data types are also provided:

- Array
- Structure

D.1.3.1 Encoding of a NULL Value

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

2) The Identifier Octet shall be present, and the class number of the NULL value shall be FAL-Specific 2.

3) The Length Octet(s) shall be omitted.

4) The Contents Octet shall be omitted.

5) The NULL type shall not be used to represent that there is no data value present. It may be used in the following two cases:

   a) An APDU which has no associated parameters:

      Example: NoEffectReqPDU ::= NULL

   b) In the CHOICE type

D.1.3.2 Encoding of a Boolean Value

1) The encoding of a Boolean value shall be primitive.

2) The Identifier Octet and Length Octet(s) shall not be present.

3) If a Boolean parameter is mandatory in an abstract syntax, then a fixed-length BitString value shall be placed where the Boolean parameter is first present in the abstract syntax and the value of the Boolean parameter and all the subsequent mandatory Boolean parameters shall be encoded in the BitString value in the order they appear. The first Boolean value shall be placed in the most significant bit of the BitString value and the following Boolean values shall be placed in the descending bits, consequently.

4) If a Boolean parameter is optional in an abstract syntax, then its value shall be encoded in the next lower order bit of the OptionalParametersMap parameter whose higher order bit is used to indicate the presence of this parameter.

5) If the Boolean value is FALSE, the corresponding bit shall be 0 (zero). If the Boolean value is TRUE, the bit shall be 1 (one).

D.1.3.3 Encoding of a Variable-Length Integer Value

1) The encoding of a variable-length Integer value shall be primitive.

2) The Identifier Octet shall not be present.

3) The Length Octet(s) shall indicate the number of octets in the Contents Octets that are actually encoded.

4) The Contents Octets 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 Contents Octets.

   NOTE The value of a two's complement binary number is derived by numbering the bits in the Contents Octets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the two's complement binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one, excluding bit 8 of the first octet, and then reducing this value by the numerical value assigned to bit 8 of the first octet if that bit is set to one.

4.1) If the Contents Octets of a variable length Integer value encoding consist of more than one octet, and if the bits of the first octet and bit 8 of the second octet are all ones or zeros, it shall be the sender's option to remove the first octet or not. This process may be repeated to eliminate all such octets. The receivers shall be able to decode the value whether or not such an octet(s) is removed.

D.1.3.4 Encoding of a Fixed-Length Integer Value

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

2) The Identifier Octet and Length Octet(s) shall not be present.

3) The Contents Octets 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 Contents Octets.

   NOTE The value of a two's complement binary number is derived by numbering the bits in the Contents Octets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the two's complement binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one, excluding bit 8 of the first octet, and then reducing this value by the numerical value assigned to bit 8 of the first octet if that bit is set to one.

D.1.3.5 Encoding of a Variable-Length Unsigned Value

1) The encoding of a variable-length Unsigned value shall be primitive.

2) The Identifier Octet shall not be present.

3) The Length Octet(s) shall indicate the number of octets in the Contents Octets that are actually encoded.

4) The Contents Octets shall be a binary number equal to the Unsigned value, and consisting 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 Contents Octets.

NOTE The value of a binary number is derived by numbering the bits in the Contents Octets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one.

4.1) If the Contents Octets of an Unsigned value encoding consist of more than one octet, and if the bits of the first octet are all zeros, it shall be the sender's option to remove the first octet or not. This process may be repeated to eliminate such octets. The receivers shall be able to decode the value whether or not such an octet(s) is removed.

D.1.3.6 Encoding of a Fixed-Length Unsigned Value

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

2) The Identifier Octet and Length Octet(s) shall not be present.

3) The Contents Octets 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 Contents Octets.

NOTE The value of a binary number is derived by numbering the bits in the Contents Octets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one.

D.1.3.7 Encoding of a Floating Point Value

1) The encoding of a Floating32 or Floating64 value shall be primitive, and the Contents Octets shall consist of exactly four or eight octets, respectively.

2) The Identifier Octet and Length Octet(s) shall not be present.

3) The Contents Octets shall contain floating point values defined in conformance with ANSI/IEEE Standard 754. The sign is encoded by using 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 and from bit 4 of the second octet for Floating64.

D.1.3.8 Encoding of a Variable-Length BitString value

1) The encoding of a variable-length BitString value shall be primitive.

2) The Identifier Octet shall not be present.

3) The Length Octet(s) shall indicate the number of octets in the BitString encoding, which is the number of octets used to encode a BitString value plus one for the "Number of Unused Bits" field (see (4)).

3.1) If the BitString is empty (no BitString value at all), there shall be no subsequent octets, and the Length Octet(s) shall be zero.

4) If the BitString is not empty, the Contents Octets shall encode the Number of Unused Bits and the BitString value as follows:

4.1) The Number of Unused Bits field shall follow the Length Octet(s) and shall always be one octet. It shall encode as an Unsigned8 type the number of unused bits in the final octet of the Contents Octets. This number shall be in the range zero to seven, inclusive.

NOTE 1 The Number of Unused Bits field is provided for the service user.

NOTE 2 The Number of Unused Bits is selected in accordance with ISO/IEC 8825.

4.2) The value of the BitString, commencing with the first bit and proceeding to the trailing bit, shall be placed in bits 8 to 1 of the first octet which follows the Number of Unused Bits field, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each octet up to and including the last octet of the Contents Octets. The unused bits shall be placed in bits 7 to 1 of the last octet, and their values are not specified.

D.1.3.9 Encoding of a Fixed-Length BitString Value

1) The encoding of a fixed-length Bitstring value of BitString8, BitString16, and BitString32 types shall be primitive, and the Contents Octets shall consist of exactly one, two, or four octets, respectively.

2) The Identifier Octet and Length Octet(s) shall not be present.

3) A BitString value, commencing with the first bit and proceeding to the trailing bit, shall be placed in 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 up to and including the last octet of the Contents Octets.

D.1.3.10 Encoding of a Variable-Length OctetString Value

1) The encoding of a variable-length OctetString value shall be primitive.

2) The Identifier Octet shall not be present.

3) The Length Octet(s) shall indicate as a binary number the number of octets in the OctetString value.

3.1) If the OctetString is of zero length, there shall be no subsequent octets, and the Length Octet(s) shall be zero.

4) If the OctetString is not of zero length, the Contents Octets shall be equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the Contents Octets.

D.1.3.11 Encoding of a Fixed-Length Octet String Value

1) The encoding of a fixed-length OctetString value of OctetString2, OctetString4, OctetString8, and OctetString16 types shall be primitive, and the Contents Octets shall consist of exactly two, four, eight, or 16 octets, respectively.

2) The Identifier Octet and Length Octet(s) shall not be present.

3) The Contents Octets shall be equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the Contents Octets.

D.1.3.12 Encoding of a Variable-Length VisibleString Value

1) The encoding of a VisibleString value shall be primitive.

2) The Identifier Octet shall not be present.

3) The Length Octet(s) shall indicate as a binary number the number of octets in the VisibleString value.

3.1) If the VisibleString is of zero length, there shall be no subsequent octets, and the Length Octet(s) shall be zero.

4) If the VisibleString is not of zero length, the Contents Octets shall be equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the Contents Octets.

D.1.3.13 Encoding of a Fixed-Length VisibleString Value

1) The encoding of a fixed-length VisibleString value of VisibleString2, VisibleString4, VisibleString8, and VisibleString16 types shall be primitive, and the Contents Octets shall consist of exactly two, four, eight, or 16 octets, respectively.

2) The Identifier Octet and Length Octet(s) shall not be present.

3) The Contents Octets shall be equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the Contents Octets.

D.1.3.14 Encoding of an ISO10646String Value

1) The encoding of an ISO10646String value shall be primitive, and the Contents Octets shall consist of zero or an even number of octets.

2) The Identifier Octet shall not be present.

3) The Length Octet(s) shall indicate as a binary number the number of octets in the Contents Octets.

3.1) If the ISO10646String is of zero length, there shall be no subsequent octets, and the Length Octet(s) shall be zero.

4) If an ISO10646String value is not of zero length, the Contents Octets shall be equal in value to the octets in the data value.

4.1) Each ISO10646 character shall be placed in two octets in the Contents Octets, with the high order byte placed in the first octet and the low-order byte in the subsequent octet, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the Contents Octets.

D.1.3.15 Encoding of a UniversalTime Value

1) The encoding of a UniversalTime value shall be primitive, and the Contents Octets shall consist of 12 octets.

2) The Identifier Octet and Length Octet(s) shall not be present.

3) The Contents Octets shall be equal in value to the octets in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the Contents Octets.

3.1) The first “D” of a UniversalTime value shall be placed in the first octet of the Contents Octets, followed by the second “D” in the second octet, up to the second “s” in the 12th octet of the Contents Octets.

3.2) A UniversalTime value consists of two sets, each of six characters. The first set is YYMMDD, where YY is the two low-order digits of the Gregorian calendar, MM is the month (counting January as 01 and December as 12), and DD is the day of the month (01 to 31). The second set is hhmmss, where hh is hour (00 to 23), mm is minutes (00 to 59), and ss is seconds (00 to 59).

3.2) In countries where the Gregorian calendar is not adapted, it is possible to substitute the Gregorian YYMM with their local calendar convention. If a local calendar system is adopted, YY shall be the two low-order digits of that calendar system and MM shall be the month counting the first month of the calendar system as 01.

D.1.3.16 Encoding of Binary Time Value

1) The encoding of a BinaryTime0, BinaryTime1, BinaryTime2, BinaryTime3, BinaryTime4, BinaryTime5, BinaryTime6, BinaryTime7, BinaryTime8 and BinaryTime9 value shall be primitive.

2) The Identifier Octet and Length Octet(s) shall not be present.

3) The Contents Octets shall be a binary number equal to the binary time value, and consisting 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 Contents Octets.

3.1) The Contents Octets of a BinaryTime0, BinaryTime1, BinaryTime2, and BinaryTime3 value shall consist of two octets.

3.2) The Contents Octets of a BinaryTime4, BinaryTime5, BinaryTime6, and BinaryTime7 value shall consist of four octets.

3.3) The Contents Octets of a BinaryTime8 and BinaryTime9 value shall consist of six octets.

NOTE The value of the granularity of each BinaryTime type is defined in IEC 61158-5.

D.1.3.17 Encoding of a CompactBCDArray Value

1) The encoding of a CompactBCDArray value shall be primitive.

2) The Identifier Octet shall not be present.

3) The Length Octet(s) shall indicate as a binary number the number of octets in the array.

3.1) If the number of BCD values is zero, there shall be no subsequent octets, and the Length Octet(s) shall be zero.

4) The Contents Octets shall encode the BCD values as the first BCD value shall be placed as a binary number in bits 8 to 5 of the first Contents Octets, the second BCD value shall be placed in bits 4 to 1 of the first Contents Octets. This will be repeated for the remaining BCD values and Contents Octets up to and including the last octet of the Contents Octets. The values of any unused bits in the last Contents Octets shall be set to F_{16}.

D.1.3.18 Encoding of an Array Value

1) An Array value shall be encoded as a SEQUENCE OF value.

D.1.3.19 Encoding of a Structure Value

1) A Structure value shall be encoded as a SEQUENCE value.

D.1.3.20 Encoding of the ANY Type

NOTE 1 The ANY type is used to convey a value of an APO, such as a data value to be written to a variable object. Since an APO is not managed by the FAL, it does not provide any encoding means for this type of value. It is assumed that the service users know the data types of such data values.

1) The encoding of the ANY type shall be primitive or constructed.

2) The Length Octet(s) shall be present and shall indicate the number of octets in the Contents Octets.

NOTE 2 This ensures that the first octet in the ANY type always indicates the total octets of the ANY type.

3) The Contents Octets shall be the encoding of a data value.

3.1) The data value shall be encoded by the service user so that the resultant encoding is identical to what is defined in this technical specification.

NOTE 3 If a data to be encoded as ANY is of type Integer, for example, the service user is responsible for providing octets that represent the value of the Integer value based on the Integer encoding rule defined in this specification.

3.2) If the encoding of a data value is primitive, the Length Octet(s) of the data value, if it is present, shall be omitted.

3.3) If an Integer value is encoded as ANY, the Contents Octets of ANY include only the Contents Octets of the Integer value. Its Length Octet(s) shall be omitted.

3.4) If the encoding of a data value is constructed, the Length Octet(s) of the data value, if it is present, shall be present in the Contents Octets of ANY.

D.1.3.21 Encoding of a BCD Value

1) A BCD value shall be encoded as an Unsigned8 value.

2) A BCD value shall be placed in bits 4 to 1 of the Contents Octets of an Unsigned8 value. The values of the bits 8 to 5 shall be zero (0).

D.1.3.22 Encoding of a FieldbusTime Value

1) A FieldbusTime value shall be encoded as a BitString32 value.

D.1.3.23 Encoding of a Compact Boolean Array Value

1) A Compact Boolean Array value shall be encoded as a BitString value.

D.1.3.24 Key words Encoding Rules

The encoding rules for the following key words are defined:

- SEQUENCE
- SEQUENCE OF
- CHOICE
- TAGGEDTYPE
- IMPLICIT
- OPTIONAL
- DEFAULT

D.1.3.24.1 Encoding of a SEQUENCE Value

1) The encoding of a SEQUENCE value shall be constructed.

2) The Identifier Octet, if it is present, shall have the class number of FAL -Specific 1.

2.1) The Identifier Octet shall be omitted from the encoding of the SEQUENCE value used in the FAL syntax descriptions specified in clause 4.

3) The Length Octet(s), if it is present, shall indicate, as a binary number, the number of productions (not the number of octets) in the SEQUENCE value to be encoded.

3.1) If there are zero data values, there shall be no subsequent octets, and the Length Octet(s) shall be zero.

3.2) The Length Octet(s) shall be omitted from the encoding of the SEQUENCE value used in the FAL syntax descriptions specified in clause 4.

NOTE Rules 2.1) and 3.2), and the fact that all the APDU abstract syntax definitions are referenced with the IMPLICIT key word, ensure that the encoding of the APDU body starts with the encoding of the first component of each APDU productions, not with the encoding of SEQUENCE or SEQUENCE OF key words that envelop the productions.

4) The Contents Octets shall consist of the complete encoding of one data value from each of the types listed in the SEQUENCE type, in the order of their appearance in the definition, unless the type was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT."

D.1.3.24.2 Encoding of a SEQUENCE OF Value

1) The encoding of a SEQUENCE OF value shall be constructed.

2) The Identifier Octet shall be present. The class number of the SEQUENCE OF value shall be FAL-Specific 1.

3) The Length Octet(s) shall indicate as a binary number the number of productions (not the number of octets) in the SEQUENCE OF value to be encoded.

3.1) If there are zero productions, there shall be no subsequent octets, and the Length Octet(s) shall be zero.

4) The Contents Octets shall consist of the complete encoding of the data values from the type listed in the SEQUENCE OF value. The order of the encoding of the data values shall be the same as the order of the data values in the SEQUENCE OF value to be encoded.

D.1.3.24.3 Encoding of Productions with OPTIONAL and DEFAULT Key Word

1) The encoding of a data value may, but need not, be present for a type which was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT."

2) If a SEQUENCE contains productions with the OPTIONAL or DEFAULT key word, those productions shall not be tagged and, in the abstract syntax constructions to be used with this encoding rule, must be placed at the end of the SEQUENCE type. Then a production

optionalParametersMap Gn_OptionalParametersMap8

or

optionalParametersMap Gn_OptionalParametersMap16

or

optionalParametersMap Gn_OptionalParametersMap32

shall be placed just before the first OPTIONAL or DEFAULT production.

3) The "optionalParametersMap" parameter, of type Gn_OptionalParametersMap8, Gn_OptionalParametersMap16, or Gn_OptionalParametersMap32 shall indicate only the OPTIONAL or DEFAULT productions which are defined in the abstract syntax.

4) The "optionalParametersMap" parameter is one of the fixed-length BitString types, and the value of one in each of the bits indicates that the corresponding OPTIONAL or DEFAULT production is present in the encoding, and the value of zero indicates that it is not present.

5) The least significant bit of the optionalParametersMap parameter corresponds to the first OPTIONAL or DEFAULT production in the SEQUENCE, the second least significant bit corresponds to the second production, and the n-th bit corresponds to the n-th production.

6) If a SEQUENCE value contains more than 32 OPTIONAL or DEFAULT productions, the BitString type shall be used as the type of the optionalParametersMap parameter.

6.1) The Number of Unused Bits field of this BitString value shall be zero since the bits are used beginning from the lsb as described above.

7) If it is possible that all the optional parameters are not present in an encoding, the optionalParametersMap parameter itself may become OPTIONAL.

7.1) When the optionalParametersMap parameter is OPTIONAL, there shall be no additional optionalParametersMap parameters present to indicate that the optionalParametersMap parameter is OPTIONAL.

D.1.3.24.4 Encoding of a CHOICE Value

1) The encoding of a CHOICE value shall be the same as the encoding of a value of the chosen type.

NOTE In the abstract syntax constructions to be used with this encoding rule, each type listed as a CHOICE must be an unambiguously tagged type, since the Identifier Octet is frequently not encoded for the type specified.

D.1.3.24.5 Encoding of a Tagged Value

1) The encoding of a Tagged value shall be primitive if the referenced encoding is primitive and constructed if it is constructed.

2) The Identifier Octet shall contain the value of the tag and its class.

2.1) The Identifier Octet for the tagged value shall not be encoded unless those used with the CHOICE type.

3) The Length Octet(s) may or may not be present.

NOTE These rules ensure that the Length Octet(s) of the Tagged value always indicates the number of octets of the Contents Octet(s) of the Tagged value.

3.1) If the referenced encoding is primitive and does not have the Length Octet(s), the Length Octet(s) of the Tagged value shall indicate the number of octets of the Contents Octet(s) of the referenced encoding.

3.2) If the referenced encoding is primitive and has the Length Octet(s), the Length Octet(s) of the referenced encoding shall be placed in the Length Octet(s) of the Tagged value.

3.3) If the referenced encoding is constructed, the Length Octet(s) of the Tagged value shall indicate the number of components of the referenced encoding.

4) The Contents Octet(s) shall be those of the referenced encoding.

D.1.3.24.6 Encoding of an IMPLICIT Value

1) The IMPLICIT keyword shall only be used with a Tagged type value, or one of the APDU class specifiers defined in the “Encoding of Type Field” clause.

2) If the IMPLICIT keyword is used, the Identifier Octet of the type which immediately follows the keyword shall not be present.

NOTE If a base encoding does not have the Identifier Octet and/or Length Octet(s), the IMPLICIT keyword does not affect the encoding.

Annex E
(normative)

Traditional Encoding Rule (TER)

E.1 Traditional Encoding Rule (TER)

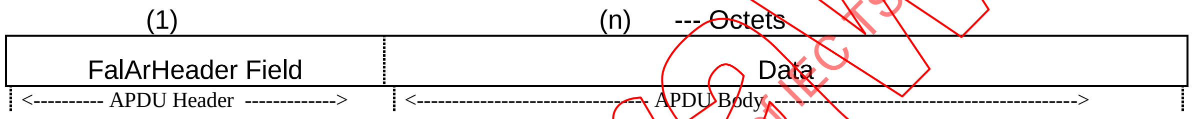
E.1.1 Introduction

The Traditional Encoding Rule (TER) is a preferable encoding rule that is compatible with existing standards.

E.1.2 TER Descriptions

E.1.2.1 Overview of Encoding

The FAL-PDUs encoded with the TER shall have a uniform format. The FAL-PDUs shall consist of two major parts, the "APDU Header" part and the "APDU Body" part as shown in the figure below:



(1)
(n) --- Octets
FalArHeader Field
Data
<---------- APDU Header ------------->
<------------------------------- APDU Body -->

Figure E.1 – APDU Overview

E.1.2.2 APDU Header Encoding

The APDU Header part is always present in all APDUs which conform to this technical specification. It consists of one field: the FalArHeader Field. Refer to annex A for the encoding rule of the FalArHeader field.

E.1.2.3 APDU Body Encoding

The FAL-PDUs are encoded with the TER in the following manner:

FAL-PDU ::= {Components}	
Components ::=	UserData || IdentificationInformation ContentsOctets
IdentificationInformation ::=	P/C Flag Tag Length
ContentsOctets ::=	OCTETSTRING

E.1.2.3.1 Structure of the Identification Information

The Identification Information consists of the P/C flag, the Tag field, and the Length field as shown in the following figure:

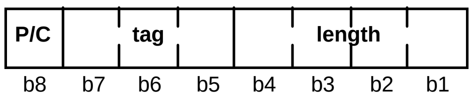


Figure E.2 – Identification Information (format 1)

The P/C flag indicates either the ContentsOctets is a simple component (primitive types, such as Integer8), or a structured component (constructed, such as SEQUENCE, SEQUENCE OF types).

P/C Flag =0 means the ContentsOctets is a simple component.
P/C Flag =1 means the ContentsOctets is a structured component.

The Tag field identifies the semantics of the ContentsOctets.

The Length field indicates the length of the ContentsOctets in octets if it is a primitive type and the number of contained components if it is a structured type.

The three components of the Identification Information are allotted in one octet as follows:

P/C flag	bit8
Tag field	bit7 - bit5, where bit7 is the msb and bit5 is the lsb.
Length field	bit3 - bit1, where bit4 is the msb and bit1 is the lsb.

If the number of the Tag is greater than six, or the number of the Length is greater than 14, the extension octet for each of the fields shall be used.

If the extension octet for the Tag is used, the Tag field in the Identification Information shall be set to seven. The Tag Extension octet that immediately follows the Identification Information shall contain the number of the Tag, with bit8 of the extension octet as the msb and bit1 the lsb as shown below:

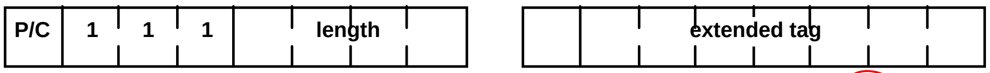


Figure E.3 – Identification Information (format 2)

If the extension octet for the Length is used, the Length field in the Identification Information shall be set to 15. The Length Extension octet that immediately follows the Identification Information shall contain the number of the Length, with bit8 of the extension octet as the msb and bit1 the lsb as shown below:

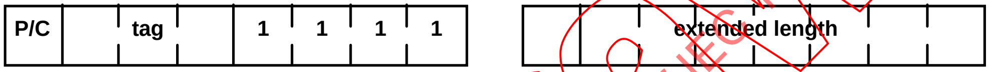


Figure E.4 – Identification Information (format 3)

If both the Tag and the Length fields are extended, the Tag Extension octet follows the Identification Information and is followed by the Length Extension octet as shown below:

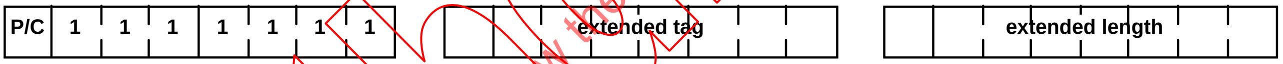


Figure E.5 – Identification Information (format 4)

If the abstract syntax does not contain a tag for a data item, then no Identification Information is encoded. The semantics and length of this data item are implicitly known by the abstract syntax.

E.1.2.4 Encoding of Simple Variables

E.1.2.4.1 Encoding of a Boolean Value

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

2) If the Boolean value is FALSE, the ContentsOctets shall be 0 (zero). If the Boolean value is TRUE, the ContentsOctets shall be FFh.

E.1.2.4.2 Encoding of an Integer Value

1) 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.

2) 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.

NOTE The value of a two's complement binary number is derived by numbering the bits in the ContentsOctets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the two's complement binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one, excluding bit 8 of the first octet, and then reducing this value by the numerical value assigned to bit 8 of the first octet if that bit is set to one.

E.1.2.4.3 Encoding of an Unsigned Value

1) 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.

2) 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.

NOTE The value of a binary number is derived by numbering the bits in the ContentsOctets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one.

E.1.2.4.4 Encoding of a Floating-Point Value

1) The encoding of a Floating32 or Floating64 value shall be primitive, and the ContentsOctets shall consist of exactly four or eight octets, respectively.

2) 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 and from bit 4 of the second octet for Floating64.

E.1.2.4.5 Encoding of a Visible String Value

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

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

2.1) If the VisibleString is of zero length, there shall be no ContentsOctets, and the Length shall be zero.

3) If the VisibleString is not of zero length, the ContentsOctets shall be equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the ContentsOctets.

E.1.2.4.6 Encoding of an Octet String Value

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

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

2.1) If the OctetString is of zero length, there shall be no ContentsOctets, and the Length shall be zero.

3) If the OctetString is not of zero length, the ContentsOctets shall be equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the ContentsOctets.

E.1.2.4.7 Encoding of a Date Value

1) The encoding of a Date value shall be primitive.

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

3) The ContentsOctets shall be equal in value to the octets in the data value, as shown in the figure below:

bits / octets	8	7	6	5	4	3	2	1	
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 E.6 – Encoding of Date Value

E.1.2.4.8 Encoding of a Time Of Day Value

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

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

3) The ContentsOctets shall be equal in value to the octets in the data value, as shown in the figure below:

bits / octets	8	7	6	5	4	3	2	1	
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 E.7 – Encoding of Time Of Day Value

E.1.2.4.9 Encoding of a Time Difference Value

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

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

3) The ContentsOctets shall be equal in value to the octets in the data value, as shown in the figure below:

bits / octets	8	7	6	5	4	3	2	1	
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 E.8 – Encoding of Time Difference Value

E.1.2.4.10 Encoding of a Bit String Value

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

2) The Length field shall indicate the number of octets in the BitString encoding, which is the number of octets used to encode a BitString.

2.1) If the BitString is empty (no BitString value at all), there shall be no ContentsOctets, and the Length field shall be zero.

3) If the BitString is not empty, the ContentsOctets shall encode the BitString value as follows:

3.1) The value of the BitString, commencing with the first bit and proceeding to the trailing bit, shall be placed in 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 up to and including the last octet of the ContentsOctets.

E.1.2.4.11 Encoding of a Time Value

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

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

3) The ContentsOctets shall be equal in value to the octets in the data value, as shown in the figure below:

bits octets	8	7	6	5	4	3	2	1	
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 bytes length of 1/32ms unit
	MSB								

Figure E.9 – Encoding of TimeValue

E.1.2.4.12 Encoding of a Null Value

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

2) The ContentsOctet shall be omitted.

E.1.2.4.13 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.

E.1.2.5 Encoding of Structured Types

When a structured type is encoded, the P/C 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.

E.1.2.5.1 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.

E.1.2.5.2 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.

E.1.2.5.3 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.

E.2 Object Definition Parameter

The mapping of the individual object definitions onto the parameters "List Of Objects and Attributes" (data type Gn_ObjectDefinition) of the GetAttributes2 and SetAttributes2 PDUs is shown in this clause.

NOTE The semantics of the Object Definition Parameter of type Gn_ObjectDefinition is application specific and the FAL does not encode or decode it. However, since prior standards defined standard semantics of this parameter, and it is used with the TER, its definition is placed here. Readers are advised that this clause is not for the FAL, but for the FAL user.

The object class 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 coded as a string of octets. In addition to the object attributes, the object class shall be transmitted in the GetAttributes and SetAttributes services, except List Header, whose numeric ID is zero.

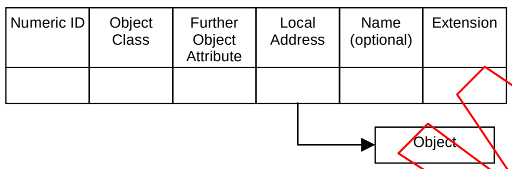


Figure E.10 – Structure of an Object Definition

```
Object-Definition ::= ListHeader
                        || DataTypeList
                        || StaticList
                        || VariableListDefinition
                        || FunctionInvocationDefinition
```

E.2.1 ListHeader

```
ListHeader ::= ANY {
    Unsigned16,                    -- numericId
    Boolean,                       -- romRamFlag
    Unsigned8,                     -- maxNameLength
    Boolean,                       -- accessProtectionSupported
    Integer16,                     -- versionOfObjectDefinition
    Unsigned32,                    -- localReferenceOfListHeader
    Unsigned16,                    -- numberOfEntriesInDataTypeList
    Unsigned32,                    -- localReferenceOfDataTypeList
    Unsigned16,                    -- firstNumericIdOfStaticList
    Unsigned16,                    -- numberOfEntriesInStaticList
    Unsigned32,                    -- localReferenceOfStaticList
    Unsigned16,                    -- firstNumericIdOfVariableListDefinition
    Unsigned16,                    -- numberOfEntriesInVariableListDefinition
    Unsigned32,                    -- localReferenceOfVariableListDefinition
    Unsigned16,                    -- firstNumericIdOfFunctionInvocationDefinition
    Unsigned16,                    -- numberOfEntriesInFunctionInvocationDefinition
    Unsigned32                     -- localReferenceOfFunctionInvocationDefinition
}
```

E.2.2 DataTypeList

```
DataTypeList ::=    DataTypeDefinition
                    ||StructuredDataTypeDefinition

DataTypeDefinition ::= ANY {
    Unsigned16,                    -- numericId
    Integer8,                      -- objectClass
    Unsigned8,                     -- dataTypeNameLength
    VisibleString                  -- dataTypeName
}
```

```
StructuredDataTypeDefinition ::= ANY {
      Unsigned16,                                    -- numericId
      Integer8,                                      -- objectClass
      Unsigned8,                                     -- numberOfElements
      recordList        SEQUENCE OF {
           Unsigned16,                               -- numericIdOfDataTypeDefinition
           Unsigned8,                                -- dataLength
      }
}
```

E.2.3 StaticList

```
StaticList ::=          VariableDefinition
      ||   ArrayDefinition
      ||   StructureDefinition
      ||   LoadRegionDefinition
      ||   EventDefinition

VariableDefinition ::= ANY {
      Unsigned16,                                    -- numericId
      Integer8,                                      -- objectClass
      Unsigned16,                                    -- numericIdOfDataTypeDefinition
      Unsigned8,                                     -- dataLength
      Vr_AccessPrivilege,                            -- accessPrivilege
      Unsigned32,                                    -- localReferenceOfVariable
      VisibleString ,                                -- variableName
      Unsigned8,                                     -- Length of the extension field. If the extension field is not present, the value
                                                        of this field shall be zero.
      OctetString                                    -- extension
}

ArrayDefinition ::= ANY {
      Unsigned16,                                    -- numericId
      Integer8,                                      -- objectClass
      Unsigned16,                                    -- numericIdOfDataTypeDefinition
      Unsigned8,                                     -- dataLength
      Unsigned8,                                     -- numberOfElements
      Vr_AccessPrivilege,                            -- accessPrivilege
      Unsigned32,                                    -- localReferenceOfArray
      VisibleString,                                 -- arrayName
      Unsigned8,                                     -- Length of the extension field. If the extension field is not present, the value
                                                        of this field shall be zero.
      OctetString                                    -- extension
}

StructureDefinition ::= ANY {
      Unsigned16,                                    -- numericId
      Integer8,                                      -- objectClass
      Unsigned16,                                    -- numericIdOfDataTypeDefinition
      Vr_AccessPrivilege,                            -- accessPrivilege
      VisibleString,                                 -- structureName
      Unsigned8,                                     -- Length of the extension field. If the extension field is not present, the value
                                                        of this field shall be zero.
      OctetString,                                   -- extension
      SEQUENCE OF Unsigned32                         -- localReferenceOfElement
}
```

```
LoadRegionDefinition ::= ANY {
      Unsigned16,                       -- numericId
      Integer8,                         -- objectClass
      Unsigned16,                       -- loadRegionSize
      Vr_AccessPrivilege,               -- accessPrivilege
      Unsigned32,                       -- localReferenceOfLoadRegion
      Unsigned8,                        -- loadRegionState
      Unsigned8,                        -- uploadState
      Integer16,                        -- numberOfRelatedObjectsInUse
      VisibleString,                    -- loadRegionName
      Unsigned8,                        -- Length of the extension field. If the extension field is not present, the value
                                           of this field shall be zero.
      OctetString                       -- extension
}

EventDefinition ::= ANY {
      Unsigned16,                       -- numericId
      Integer8,                         -- objectClass
      Unsigned16,                       -- numericIdOfEventData
      Unsigned8,                        -- eventDataLength
      Vr_AccessPrivilege,               -- accessPrivilege
      Boolean,                          -- enabled
      VisibleString,                    -- eventName
      Unsigned8,                        -- Length of the extension field. If the extension field is not present, the value
                                           of this field shall be zero.
      OctetString                       -- extension
}
```

E.2.4 VariableListDefinition

```
VariableListDefinition ::= ANY {
      Unsigned16,                       -- numericId
      Integer8,                         -- objectClass
      Unsigned16,                       -- numericIdOfElements
      Vr_AccessPrivilege,               -- accessPrivilege
      Boolean,                          -- deletable
      SEQUENCE OF Unsigned32,
      VisibleString,                    -- variableListName
      Unsigned8,                        -- Length of the extension field. If the extension field is not present, the value
                                           of this field shall be zero.
      OctetString                       -- extension
}
```

E.2.5 FunctionInvocationDefinition

```
FunctionInvocationDefinition ::= ANY {
      Unsigned16,                       -- numericId
      Integer8,                         -- objectClass
      Unsigned8,                        -- numberOfRelatedObjects
      Fi_AccessPrivilege,               -- accessPrivilege
      Boolean,                          -- deletable
      Boolean,                          -- reusable
      Unsigned8,                        -- functionInvocationState
      SEQUENCE OF Unsigned16,           -- NumericIdOfLoadRegion
      VisibleString,                    -- functionInvocationName
      Unsigned8,                        -- Length of the extension field. If the extension field is not present, the value
                                           of this field shall be zero.
      OctetString                       -- extension
}
```

Annex F
(normative)

Buffer-Oriented Encoding Rules (BER) and Messaging Encoding Rule (MER)

F.1 Encoding Rule for Buffer Services (BER)

F.1.1 Introduction

The Compact Encoding Rules (BER) is an encoding rule for new implementation where Time critical transfer is selected.

F.1.2 Application Layer Encoding Rules

F.1.2.1 Overview of Encoding

The PDUs that conform to this technical specification shall be encoded in a uniform format as shown in the figure below. The PDUs consist of two major parts: the "APDU Header" part and the "APDU Body" part.

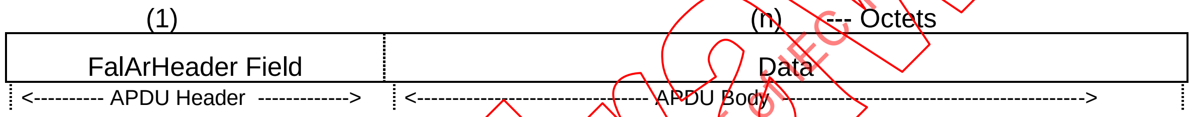

(1)
(n) --- Octets
FalArHeader Field
Data
<---------- APDU Header ------------->
<-------------------------------- APDU Body -->

Figure F.1 – APDU Overview

F.1.2.2 APDU Header Encoding

The APDU Header part is always present in all APDUs which conform to this part of ISA-S50.02. It consists of one field: the FalArHeader Field. Refer to annex A for the encoding rule of the FalArHeader field.

F.1.2.3 APDU Body Encoding

The FAL encoding rules are based on terms and conventions defined in ISO/IEC standards. The encoding consists of three components in the following order:

- An Identifier Octet
- Length Octet(s)
- The ContentsOctets

F.1.2.3.1 Identifier Octet

1) The Identifier Octet shall encode the type and shall consist of one octet.

2) The value of this octet is 40 h. Other values are reserved for management.

F.1.2.3.2 Length Octet

1) The Length Octet shall consist of one octet.

F.1.2.3.3 ContentsOctets

1) The ContentsOctets shall encode the data value as a string of octet.

F.2 Encoding Rule for Messaging Services (MER)

The encoding of an ASN.1 type value may comprise the following components:

- Identification Octet
- Content Length Octets
- Content Octets

F.2.1 Identification Octet

The Identification Octet is used for the encoding of the type tag associated with the value. It contains the class and number of the type tag.

The Identification Octet only exists in the encoding of tagged types appearing in a CHOICE. If present, the Identification Octet is structured as follows:

• bit 8 defines the value type class:

Table F.1 – Identification Octet Classes

class	bit 8 value
FAL specific	0
Context-specific	1

• bits 7 to 1 are used for encoding the identifier numbers in the range 0 to 127, in the form of an unsigned binary number, whose most and least significant bits are respectively bits 7 and 1.

F.2.2 Contents length octets

If present, the encoding of the contents length of the type value always consists of two octets representing a number of units of measurement (unsigned binary number).

The unit of measurement is type-dependent, as follows:

Table F.2 – Unit of Measurement of Contents Length Octets

SEQUENCE OF	octet
SEQUENCE	octet
INTEGER	octet
BOOLEAN	octet
BIT STRING	bit
OCTET STRING	octet

F.2.3 ContentsOctets

The ContentsOctets shall encode the data value according to the encoding rules defined for the respective type as specified in the following subclauses. The ContentsOctets may consist of zero, one or more octets.

F.3 Type Encoding Rules

F.3.1 Boolean

A Boolean value shall be encoded as follows:

- The Identifier Octet and the Length Octets shall not be present.
- The ContentsOctets always consist of one byte. If the Boolean value equals FALSE, all bits of the octet are 0. If the Boolean value equals TRUE, the octet can contain any combination of bits other than the encoding for FALSE.

F.3.2 Integer

An integer value shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present, if the size of the integer type is invariable. An integer with invariable size is created by constraining the possible value. The length octets shall be present, if the size of the integer value is variable.

he ContentsOctets shall contain the two complement binary number equal to the integer value. The most significant bits of the integer value are encoded in bit8 to bit1 of the first octet, the next in bit8 to bit1 of the next octet and so on. If the values of an integer type are restricted to negative and non-negative numbers, bit8 of the first octet gives the sign of the value, if the values are restricted to non-negative numbers only, no sign bit is needed (see examples below)

Examples:

- INTEGER (-128..128) is an integer with invariable octet length of one.
- INTEGER (0..255) is also an integer with invariable octet length of one.
- INTEGER is an integer with variable octet length.

F.3.3 Bit string

A bitstring value shall be encoded as follows:

- The Identifier Octet shall not be present
- The Length Octets shall not be present, if the size of the bitstring type is invariable. A bitstring with invariable size is created by applying a size constraint containing only one value on the bitstring type. The length octets shall be present, if the size of the bitstring value is variable.
- The ContentsOctets comprise as many octets as necessary to contain all bits of the actual value: N_Octets = (N_Bits-1) div 8 + 1. The bitstring value commencing with the first bit and proceeding to the trailing bit, shall be placed in bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet and so on. If the number of bits is not a multiple of 8 there are so-called unused bits, which are located in the least significant bits of the last octet. The value of the unused bits may be zero or one and carry no meaning.

Examples:

- BIT STRING SIZE (30) is a bitstring with invariable octet length of four and two unused bits.
- BIT STRING SIZE (1..32) is a bitstring with variable octet length.
- BIT STRING is a bitstring with variable octet length.

F.3.4 Octet string

An octetstring value shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present, if the size of the octetstring type is invariable. An octetstring with invariable size is created by applying a size constraint containing only one value on the octetstring type. The length octets shall be present, if the size of the octetstring value is variable.
- The ContentsOctets shall be equal in value to the octets in the data value.

Examples:

- OCTET STRING SIZE (30) is an octetstring with invariable octet length of 30.
- OCTET STRING SIZE (1..32) is an octetstring with variable octet length.
- OCTET STRING is an octetstring with variable octet length.

F.3.5 Visible string

A visiblestring value shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present, if the size of the visiblestring type is invariable. A visiblestring with invariable size is created by applying a size constraint containing only one value on the visiblestring type. The length octets shall be present, if the size of the visiblestring value is variable.
- The ContentsOctets shall be equal in value to the octets in the data value.

Examples:

- VisibleString SIZE (30) is a visiblestring with invariable octet length of 30.
- VisibleString SIZE (1..32) is a visiblestring with variable octet length.
- VisibleString is a visiblestring with variable octet length.

F.3.6 SEQUENCE Types

A value of a SEQUENCE type shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall be there and specify the number of octets being used for the ContentsOctets. However, for the first Keyword “SEQUENCE” of FalArPDU, this length shall not be encoded at all.
- The ContentsOctets shall consist of the encoding of all the element types in the same order as they are specified in the ASN.1 description of the SEQUENCE type.

F.3.7 SEQUENCE OF Types

A value of a SEQUENCE OF type shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall be there and specify the number of octets being used for the ContentsOctets.
- The ContentsOctets shall consist of the encoding of all the element types in the same order as they are specified in the ASN.1 description of the SEQUENCE OF type.

F.3.8 CHOICE Types

A value of a CHOICE type shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present.
- The ContentsOctets shall consist of the encoding of the selected type of the alternative type list.

F.3.9 Null

A value of a NULL type shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present.
- The ContentsOctets shall not be present.

F.3.10 Tagged Types

A value of a tagged type shall be encoded as follows:

- The Identifier Octet shall only be present, if the tagged type is part of an Alternative Type List in a CHOICE construct.
- The Length Octets shall not be present.
- The ContentsOctets shall consist of the encoding of the Type which was tagged.

F.3.11 IMPLICIT Types

A value of an IMPLICIT type shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present.
- The ContentsOctets shall consist of the encoding of the Type being referenced by the IMPLICIT construct, except for the case when the referenced Type is a SEQUENCE type. In this case, the ContentsOctets consist only of the ContentsOctets of the referenced SEQUENCE type, the length information of this SEQUENCE type should not occur in the encoding.

F.3.12 OPTIONAL and DEFAULT Types

For the encoding of a value of an OPTIONAL or DEFAULT type there are two possibilities:

1) If the OPTIONAL/DEFAULT type is a subtype of a SEQUENCE type containing a Gn_OptionalParametersMap element, which is used to determine the existence or non-existence of all optional subtypes within this structure. The least significant bit of this Gn8OptionalParametersMap type corresponds to the first OPTIONAL or DEFAULT subtype, the second least significant bit corresponds to the second subtype and so on.

In this case, a value of an OPTIONAL or DEFAULT type shall be encoded as follows:

If the corresponding bit of the Gn_OptionalParametersMap is zero or the Gn_OptionalParametersMap is missing altogether (it may be an OPTIONAL itself), the OPTIONAL/DEFAULT value does not exist and should not be encoded at all. If the corresponding bit of the Gn_OptionalParametersMap is one, the value exists, and its encoding should be as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present.
- The ContentsOctets shall consist of the encoding of the Type being referenced by the OPTIONAL/DEFAULT construct.

2) Otherwise, a value of an OPTIONAL or DEFAULT type shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall be present. If there is no value for this type, the Length Octets contain the value 0.
- The ContentsOctets shall consist of the encoding of the referenced Type, if there is a value for this type, otherwise no ContentsOctets exist.

If a Gn_OptionalParametersMap type is defined as OPTIONAL itself, it is always encoded in the second way.

F.3.13 ANY Types

An ANY type is used for the definition of complex types, whose structure is described informally rather than in ASN.1, generally because its definition in ASN.1 would not result in an optimal encoding. Usually the structuring information (Identifier Octets, Length Octets) in the encoding of such types can be omitted, since the receiver of the encoded type has additional knowledge, which enables him to encode such an optimized PDU.

A value of an ANY type shall be encoded as follows:

- The Identifier Octet shall not be present.
- The Length Octets shall not be present.
- The ContentsOctets shall consist of the encoding of all implicit types which constitute the ANY type.

F.3.14 Encoding of APDU Header

The APDU Header always begins with the FALArHeader. This implies that the Contents Length of the surrounding SEQUENCE is not encoded.

Annex G
(normative)

Queued Usertriggered Unidirectional (QUU) ARPM

G.1 Primitive Definitions

G.1.1 Primitives Exchanged between ARPM and FSPM

Table G.1 – 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	remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) request primitive from the FSPM to the ARPM.

Table G.2 – 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.
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, remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) indication primitive from the ARPM to the FSPM.
FSTS_ind	ARPM	arep_id, reported_status	This is an FAL internal primitive used to convey confirmation status.

G.2 Parameters of FSPM/ARPM Primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in table G.3.

Table G.3 – 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 specification.
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.
remote_dlsap_address	This parameter conveys value that is used for the Remote_DLSAP_Address parameter.

G.2.1 DLL Mapping of QUU AREP Class

This clause describes the mapping of the QUU 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 specification; rather, it defines how they are used by this AR class. A means to configure and monitor the values of these attributes will be provided in the future IEC 61158-7.

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

CLASS: QuuCl

PARENT CLASS: QueuedUser-TriggeredUnidirectionalAREP

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlsapAddress
2	(c)	Constraint:	RemoteAddressConfigurationType = Linked
2.1	(m)	Attribute:	RemoteDlsapAddress
3	(m)	Attribute:	LocalDlsapRole (Basic, Group)
4	(c)	Constraint:	Role = ReportSource
4.1	(m)	Attribute:	DefaultQosAsSender
4.1.1	(m)	Attribute	DllPriority (Urgent, Normal, TimeAvailable)
4.1.2	(m)	Attribute:	MaxConfirmDelayOnUnitdata
4.1.3	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
4.1.4	(m)	Attribute:	DlSchedulingPolicy (Implicit)
4.1.5	(o)	Attribute:	DleRemoteConf (True, False)
5	(m)	Attribute:	ExplicitQueue (True, False)
6	(c)	Constraint:	ExplicitQueue = True
6.1	(m)	Attribute:	QueueBindings—either for a sender or a receiver
6.1.1	(m)	Attribute:	MaxQueueDepth
6.1.2	(m)	Attribute:	MaxDlsduSize

DLL SERVICES:

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

G.2.1.1 Attributes

G.2.1.1.1 LocalDlsapAddress

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

G.2.1.1.1.1 RemoteDlsapAddress

This attribute specifies the remote address to which FAL-PDUs are sent (for ReportSource AREPs), or from which they are received (for ReportSink AREPs).

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

G.2.1.1.2 LocalDlsapRole

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

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

G.2.1.1.3 DefaultQosAsSender

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

G.2.1.1.3.1 DllPriority

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

This attribute supplies the value for the “DLL priority” parameter of the DLL. The values Urgent, Normal, and Time-Available correspond to URGENT, NORMAL, and TIME-AVAILABLE as defined in the Fieldbus Data Link Layer specification, respectively.

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

G.2.1.1.3.2 MaxConfirmDelayOnUnitdata

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

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

G.2.1.1.3.3 DlpduAuthentication

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

This attribute supplies the value for the “DLPDU authentication” parameter of the DLL. The values Ordinary, Source, and Maximal correspond to ORDINARY, SOURCE, and MAXIMAL as defined in the Fieldbus Data Link Layer specification, respectively.

G.2.1.1.3.4 DlSchedulingPolicy

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

This attribute supplies the value for DL-Scheduling-policy attribute. Only the value Implicit is used. It corresponds to IMPLICIT as defined in the Fieldbus Data Link Layer specification.

G.2.1.1.3.5 DleRemoteConf

This optional attribute specifies, when it is present and true, that the remote immediate DLL acknowledgement shall be used. If it is not present, the remote immediate DLL acknowledgement shall not be used.

G.2.1.1.4 ExplicitQueue

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

G.2.1.1.4.1 QueueBindings

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

G.2.1.1.4.2 MaxQueueDepth

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

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

G.2.1.1.4.3 MaxDlsduSize

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

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

G.2.1.2 DLL Services

Refer to annex C, Data Link Layer Service Selection, for DLL service descriptions.

G.3 QUU ARPM State Machine

G.3.1 QUU ARPM States

The defined states and their descriptions of the QUU ARPM are listed below:

Table G.4 – QUU ARPM States

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.

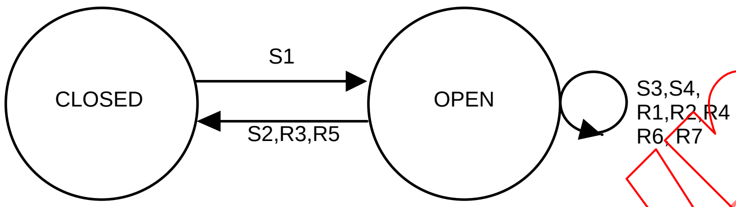


Figure G.1 – State Transition Diagram of the QUU ARPM

G.3.2 QUU ARPM state table

Table G.5 – QUU ARPM State Table - Sender Transactions

#	Current State	Event Action	Next State

S1	CLOSED	EST_req
		
		EST_cnf(+) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			OPEN

S2	OPEN	Abort_req (no actions taken)	CLOSED

S3	OPEN	UCS_req
		&& RemoteAddressConfigurationType = "Linked"
		&& Role = "ReportSource"
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "UCS_PDU",
		fal_data := user_data)
		}
			OPEN

S4	OPEN	UCS_req
		&& RemoteAddressConfigurationType = "Free"
		&& Role = "ReportSource"
		
		RemoteDlsapAddress := remote_dlsap_address,
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "UCS_PDU",
		fal_data := user_data)
		}
			OPEN

Table G.6 – QUU ARPM State Table - Receiver Transactions

#	Current State	Event Action	Next State

R1	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Unitdata_ind”
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlsapAddress = calling_address
		&& Role = “ReportSink”
		&& FAL_Pdu_Type (fal_pdu) = “UCS_PDU”
		
		UCS_ind {
		arep_id := GetArepId (),
		user_data := fal_pdu
		}
			OPEN

R2	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Unitdata_ind”
		&& RemoteAddressConfigurationType = “Free”
		&& Role = “ReportSink”
		&& FAL_Pdu_Type (fal_pdu) = “UCS_PDU”
		
		UCS_ind {
		arep_id := GetArepId (),
		remote_dlsap_address := calling_address,
		user_data := fal_pdu
		}
			OPEN

R3	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Unitdata_ind”
		&& Role = “ReportSink”
		&& FAL_Pdu_Type (fal_pdu) <> “UCS_PDU”
		
		Abort_ind {
		arep_id := GetArepId (),
		locally_generated := "True",
		identifier := "FAL",
		reason_code := " Invalid FAL-PDU",
		additional_detail := "null"
		}
			CLOSED

R4	OPEN	ErrorToARPM (no actions taken)	OPEN

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

R6	OPEN	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_unidata_cnf"
		&& Role = “ReportSource”
		&& Fal_Pdu_Type(fal_pdu) = "UCS_PDU"
		&& reason = "success"
		
		FSTS_Ind{
		arep_id := GetArep(),
		reported_status := “dl_unidata_cnf”
		}
			OPEN

#	Current State	Event Action	Next State

R7	OPEN	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_unidata_cnf"
		&& Role = “ReportSource”
		&& Fal_Pdu_Type(fal_pdu) = "UCS_PDU"
		&& reason <> "success"
		
		FSTS_Ind{
		arep_id := GetArep(),
		reported_status := “unidata_cnf NOACK”
		}
			OPEN

G.3.3 Functions used by QUU ARPM

Table G.7 – Function GetArepId ()

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

Table G.8 – Function BuildFAL-PDU

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

Table G.9 – Function FAL_Pdu_Type

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

Annex H
(normative)

Queued Usertriggered Bidirectional-Connection Oriented (QUB-CO) ARPM

H.1 Primitive Definitions

H.1.1 Primitives Exchanged between ARPM and FSPM

Table H.1 – Primitives issued by FSPM to ARPM

Primitive Name	Source	Associated parameters	Functions
EST_req	FSPM	user_data, remote_dlcep_address	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.

Table H.2 – 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.

H.1.2 Parameters of FSPM/ARPM Primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in table H.3.

Table H.3 – 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 specification.
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.
reported_status	This parameter conveys a Data Link Layer event status.
Remote_dlcep_address	This parameter conveys value of the remote dlcep address.

H.2 DLL Mapping of QUB AREP Class

This clause describes the mapping of the QUB 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 specification; rather, it defines how they are used by this AR class. A means to configure and monitor the values of these attributes will be provided in the future IEC 61158-7.

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

CLASS: QubCo

PARENT CLASS: QueuedUser-TriggeredBidirectionalAREP

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(c)	Constraint:	RemoteAddressConfigurationType = Linked
2.1	(m)	Attribute:	RemoteDlcepAddress
3	(c)	Constraint:	Role = Client || Peer
3.1	(m)	Attribute:	Initiator (True, False)
4	(m)	Attribute:	DlsapRole (Basic)
5	(m)	Attribute:	QosParameterSet
5.1	(m)	Attribute:	DlcepClass (Peer)
5.2	(m)	Attribute:	DlcepDataDeliveryFeatures
5.2.1	(m)	Attribute:	FromRequestorToResponder (Classical, Disordered)
5.2.2	(m)	Attribute:	FromResponderToRequestor (Classical, Disordered)
5.3	(m)	Attribute:	Priority
5.3.1	(m)	Attribute:	DllPriority (Urgent, Normal, TimeAvailable)
5.3.2	(m)	Attribute:	DllPriorityNegotiated (Urgent, Normal, TimeAvailable)
5.4	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
5.5	(m)	Attribute:	ResidualActivity
5.5.1	(m)	Attribute:	ResidualActivityAsSender (True, False)
5.5.2	(m)	Attribute:	ResidualActivityAsReceiver (True, False)
5.6	(m)	Attribute:	MaxConfirmDelay
5.6.1	(m)	Attribute:	MaxConfirmDelayOnDlConnect
5.6.2	(m)	Attribute:	MaxConfirmDelayOnDlData
5.7	(m)	Attribute:	DlSchedulingPolicy (Implicit)
5.8	(m)	Attribute:	ExplicitQueue (True, False)
5.9	(c)	Constraint:	ExplicitQueue = True
5.9.1	(m)	Attribute:	MaxDlsduSizes
5.9.1.1	(m)	Attribute:	MaxDlsduSizeFromRequestor
5.9.1.2	(m)	Attribute:	MaxDlsduSizeFromResponder
5.9.1.3	(m)	Attribute:	MaxDlsduSizeFromRequestorNegotiated
5.9.1.4	(m)	Attribute:	MaxDlsduSizeFromResponderNegotiated
5.9.2	(m)	Attribute:	MaxQueueDepth
5.9.2.1	(m)	Attribute:	MaxSendingQueueDepth
5.9.2.2	(m)	Attribute:	MaxReceivingQueueDepth

DLL SERVICES:

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

H.2.1 Attributes

H.2.1.1 LocalDlcepAddress

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

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

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

H.2.1.2 RemoteAddressConfigurationType

This attribute specifies how this AREP is used with the remote AREP. The value of “Free” means that this AREP can communicate with any remote AREP. The value of “Linked” means that this AREP can only communicate with the AREP specified by the RemoteDlcepAddress attribute.

H.2.1.2.1 RemoteDlcepAddress

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

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

H.2.1.3 Role

This attribute specifies possible roles of this AREP. The value of “Client” means that this AREP issues FAL request PDUs to a remote AREP whose Role is either “Server” or “Peer.” An AREP whose Role is “Server” is a mirror of “Client.” An AREP whose Role is “Peer” can act as a “Client” and a “Peer” simultaneously.

H.2.1.3.1 Initiator

This attribute specifies whether this AREP is capable of issuing a DL-Connect request or not. The value of “True” means it is capable of doing it. If the value of this attribute is “False,” it can only accept a DL-Connect indication from a remote AREP.

H.2.1.4 DlsapRole

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

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

H.2.1.5 QosParameterSet

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

H.2.1.5.1 DlcepClass

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

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

H.2.1.5.2 DlcepDataDeliveryFeatures

These two attributes specify data delivery features of the DLL.

The permitted values Classical and Disordered correspond respectively to CLASSICAL and DISORDERED defined by the Data Link Layer specification.

The FromRequestorToResponder and FromResponderToRequestor attributes shall have the same value.

H.2.1.5.2.1 FromRequestorToResponder

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

H.2.1.5.2.2 FromResponderToRequestor

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

H.2.1.5.3 Priority

H.2.1.5.3.1 DllPriority

This attribute specifies the configured value of the DLL priority.

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

H.2.1.5.3.2 DllPriorityNegotiated

This attribute specifies the negotiated value of the DLL priority.

H.2.1.5.4 DlpduAuthentication

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

This attribute supplies the value for the “DLPDU-authentication” parameter of the DLL. The permitted value Ordinary, Source, and Maximal correspond to ORDINARY, Source, and MAXIMAL, respectively, as defined in the Fieldbus Data Link Layer specification.

H.2.1.5.5 ResidualActivityAsSender

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

H.2.1.5.6 ResidualActivityAsReceiver

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

H.2.1.5.7 MaxConfirmDelay

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

H.2.1.5.7.1 MaxConfirmDelayOnDlConnect

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

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

H.2.1.5.7.2 MaxConfirmDelayOnDlData

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

This attribute supplies the value for the “Maximum confirmation delay on DL-Data” parameter specified in the Data Link Layer specification (IEC 61158-3).

H.2.1.5.8 DlSchedulingPolicy

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

This attribute supplies the value for the “DL-Scheduling-policy” parameter of the DLL. The permitted value Implicit corresponds to IMPLICIT defined in the Fieldbus Data Link Layer specification.

H.2.1.5.9 ExplicitQueue

H.2.1.5.9.1 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.

H.2.1.5.9.2 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.

H.2.1.5.9.3 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.

H.2.1.5.9.4 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.

H.2.1.5.9.5 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.

H.2.1.5.9.6 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.

H.2.2 DLL Services

Refer to annex C for DLL service descriptions.

H.3 QUB AREP State Machine

H.3.1 QUB ARPM States

The defined states and their descriptions of the QUB ARPM are listed below:

Table H.4 – QUB ARPM States

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.ind 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.

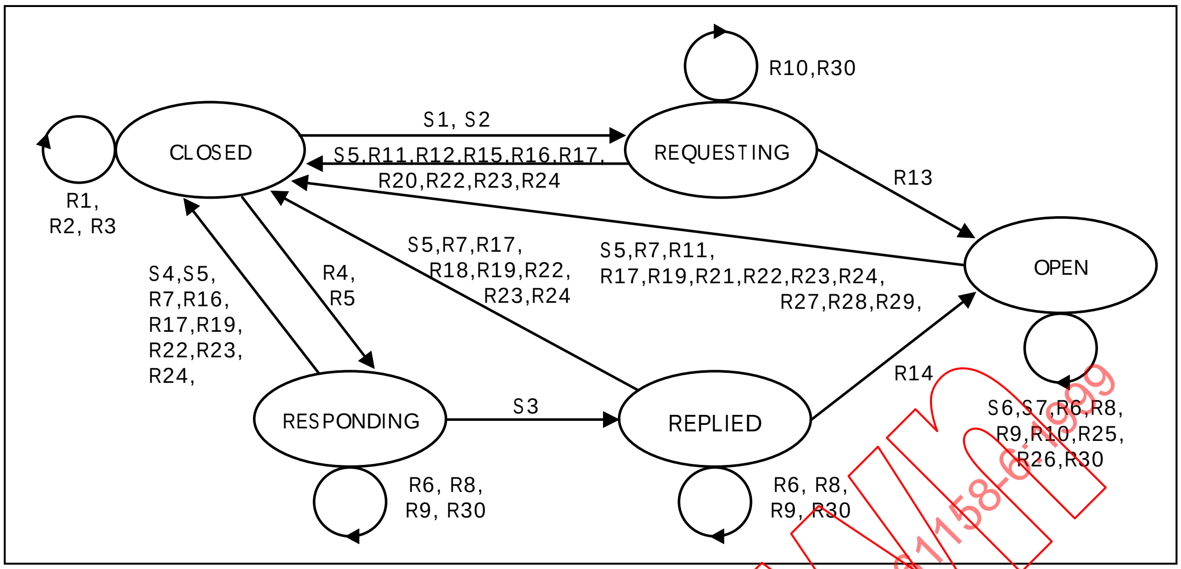

CLOSED
REQUESTING
OPEN
RESPONDING
REPLIED
R1, R2, R3
S1, S2
S5,R11,R12,R15,R16,R17, R20,R22,R23,R24
R10,R30
R13
S4,S5, R7,R16, R17,R19, R22,R23, R24,
R4, R5
S5,R7,R17, R18,R19,R22, R23,R24
S5,R7,R11, R17,R19,R21,R22,R23,R24, R27,R28,R29,
R14
S6,S7,R6,R8, R9,R10,R25, R26,R30
S3
R6, R8, R9, R30
R6, R8, R9, R30

Figure H.1 – State Transition Diagram of QUB ARPM

H.3.2 QUB ARPM state table

Table H.5 – QUB ARPM State Table - Sender Transactions

#	Current State	Event 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

#	Current State	Event Action	Next State
S3	RSP	EST_rsp(+)	REPL
		FAL-PDU_req {	
		dmpm_service_name := "DMPM_Connect_rsp",	
		arep_id := GetArepId (),	
		responding_address := "default dlsap address",	
		local_dlcep_address := LocalDlcepAddress,	
		dlsdu := BuildFAL-PDU (	
		fal_pdu_name := "EST_RspPDU",	
		fal_data := user_data)	
		}	
S4	RSP	EST_rsp(-)	CLOSED
		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)	
		}	
S5	NOT CLOSED	Abort_req	CLOSED
		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)	
		}	
S6	OPEN	CS_req	OPEN
		&& Role = "Client" || "Peer"	
		FAL-PDU_req {	
		dmpm_service_name := "DMPM_Data_req",	
		arep_id := GetArepId (),	
		dlsdu := BuildFAL-PDU (	
		fal_pdu_name := "CS_ReqPDU",	
		fal_data := user_data)	
		}	
S7	OPEN	CS_rsp	OPEN
		&& Role = "Server" || "Peer"	
		FAL-PDU_req {	
		dmpm_service_name := "DMPM_Data_req",	
		arep_id := GetArepId (),	
		dlsdu := BuildFAL-PDU (	
		fal_pdu_name := "CS_RspPDU",	
		fal_data := user_data)	
		}	

Table H.6 – QUB ARPM State Table - Receiver Transactions

#	Current State	Event Action	Next State

R1	CLOSED	Connect_ind
		&& Initiator = “True”
		
		FAL-PDU_req {
		dmpm_service_name := “DMPM_Disconnect_req”,
		arep_id := GetArepId (),
		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 (),
		reason := “Invalid FAL-PDU”,
		dlsdu := “null”
		}
			CLOSED

R3	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 (),
		reason := "Remote Address Mismatch",
		dlsdu := “null”
		}
			CLOSED

R4	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ReqPDU”
		&& 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

R5	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_Type (dls_user_data) = “EST_ReqPDU”
		&& 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 Action	Next State

R6	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 (),
		reason := "Remote Address Mismatch",
		dlsdu := “null”
		}
			SAME

R7	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& RemoteDlcepAddress = calling_dlcep_address
		
		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

R8	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 (),
		reason := "AREP Busy",
		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 (),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		}
			SAME

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

#	Current State	Event Action	Next State

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

R12	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

R13	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

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

R15	REQ	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& FAL_Pdu_Type (fal_pdu) = “EST_ErrPDU”
		
		EST_cnf(-) {
		arep_id := GetArepId (),
		user_data := fal_pdu
		}
			CLOSED

#	Current State	Event Action	Next State

R16	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

R17	NOT CLOSED	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& fal_pdu <> “null”
		&& 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

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	REPL
	RSP
	OPEN
		FAL-PDU_ind
		&& 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 Action	Next State

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

R21	OPEN	FAL-PDU_ind
		&& ((dmpm_service_name <> "DMPM_Disconnect_ind")
		&& (dmpm_service_name <> "DMPM_Data_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

R22	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"
		}
			CLOSED

R23	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"
		}
			CLOSED

#	Current State	Event Action	Next State

R24	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"
		}
			CLOSED

R25	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Peer” || “Server”
		&& FAL_Pdu_Type (fal_pdu) = “CS_ReqPDU”
		
		CS_ind {
		arep_id := GetArepId (),
		user_data := fal_pdu
		}
			OPEN

R26	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Client” || “Peer”
		&& FAL_Pdu_Type (fal_pdu) = “CS_RspPDU”
		
		CS_cnf {
		arep_id := GetArepId (),
		user_data := fal_pdu
		}
			OPEN

R27	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Server”
		&& FAL_Pdu_Type (fal_pdu) <> “CS_ReqPDU”
		
		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"
		}
			CLOSED

#	Current State	Event Action	Next State

R28	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Client”
		&& FAL_Pdu_Type (fal_pdu) <> “CS_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”,
		additional_detail := "null"
		}
			CLOSED

R29	OPEN	FAL-PDU_ind
		&& Role = “Peer”
		&& dmpm_service_name = “DMPM_Data_ind”
		&& ((FAL_Pdu_Type (fal_pdu) <> “CS_ReqPDU”)
		&& (FAL_Pdu_Type (fal_pdu) <> “CS_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”,
		additional_detail := "null"
		}
			CLOSED

R30	NOT CLOSED	ErrorToARPM (No actions taken. See NOTE below.) 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

H.3.3 Functions used by QUB ARPM

Table H.7 – Function GetArepId ()

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

Table H.8 – 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 H.9 – Function FAL_Pdu_Type

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

Table H.10 – 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 H.11 – 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 H.12 – 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.			

Annex I
(normative)

Queued Usertriggered Bidirectional-Connectionless (QUB-Cl) ARPM

I.1 Primitive Definitions

I.1.1 Primitives Exchanged between ARPM and FSPM

Table I.1 – 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.
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.
UCS_req	FSPM	remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) request primitive from the FSPM to the ARPM.

Table I.2 – 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.
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.
UCS_ind	ARPM	arep_id, remote_dlsap_address, 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.
FSTS_ind	ARPM	arep_id, reported_status	This is an FAL internal primitive used to convey a FAL-Status (FSTS) indication primitive from the ARPM to the FSPM.

I.1.2 Parameters of FSPM/ARPM Primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in table I.3.

Table I.3 – 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 specification.
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.
duplicate_fal_sdu	This parameter conveys value that is used for the Duplicate_FAL-SDU parameter.
remote_dlsap_address	This parameter conveys value that is used for the Remote_DLSAP_Address parameter.
status	This parameter conveys value that is used for the Status parameter.
reported_status	This parameter conveys a Data Link Layer event status.
local_timeliness	This parameter conveys value that is used for the Local_Timeliness parameter.
remote_timeliness	This parameter conveys value that is used for the Remote_Timeliness parameter.

I.2 DLL Mapping of QUB-CL AREP Class

This clause describes the mapping of the QUB_CL 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 specification; rather, it defines how they are used by this AR class. A means to configure and monitor the values of these attributes will be provided in the future IEC 61158-7.

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

CLASS: QubCl

PARENT CLASS: QueuedUser-TriggeredBidirectionalConnectionlessAREP

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlsapAddress
2	(m)	Attribute:	RemoteDlsapAddress
3	(m)	Attribute:	LocalDlsapRole (Basic)
4	(m)	Attribute:	DefaultQosAsSender
4.1	(m)	Attribute	DllPriority (Urgent, Normal, TimeAvailable)
4.2	(m)	Attribute:	MaxConfirmDelayOnUnitdata
4.3	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
4.4	(m)	Attribute:	DlSchedulingPolicy (Implicit)
4.5	(m)	Attribute:	DleRemoteConf (True, False)
5	(m)	Attribute:	ExplicitQueue (True, False)
6	(c)	Constraint:	ExplicitQueue = True
6.1	(m)	Attribute:	SendingQueueBindings
6.1.1	(m)	Attribute:	MaxQueueDepth
6.1.2	(m)	Attribute:	MaxDlsduSize
6.2	(m)	Attribute:	ReceivingQueueBindings
6.2.1	(m)	Attribute:	MaxQueueDepth
6.2.2	(m)	Attribute:	MaxDlsduSize

DLL SERVICES:

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

I.2.1 Attributes

I.2.1.1 LocalDlsapAddress

This attribute specifies the DLSAP address to which this AREP is attached. It supplies the value for the "DL(SAP)-address" parameter specified in the DLL.

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

I.2.1.2 RemoteDlsapAddress

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

I.2.1.3 LocalDlsapRole

This attribute specifies the behavior of the local DLSAP to be used. No DLCEPs are needed on this Basic DLSAPRole.

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

I.2.1.4 DefaultQosAsSender

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

I.2.1.4.1 DllPriority

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

This attribute supplies the value for the “DLL priority” parameter of the DLL. The values Urgent, Normal, and Time-Available correspond to URGENT, NORMAL, and TIME-AVAILABLE as defined in the Fieldbus Data Link Layer specification, respectively.

NOTE It is not possible to use different priorities for each FAL-PDU sent from the same QUB_CL AREP. The priority is identical on the sending way and on the receiving way.

I.2.1.4.2 MaxConfirmDelayOnUnitdata

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

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

I.2.1.4.3 DlpduAuthentication

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

This attribute supplies the value for the “DLPDU authentication” parameter of the DLL. The values Ordinary, Source, and Maximal correspond to ORDINARY, SOURCE, and MAXIMAL as defined in the Fieldbus Data Link Layer specification, respectively.

I.2.1.4.4 DlSchedulingPolicy

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

This attribute supplies the value for DL-Scheduling-policy attribute. Only the value Implicit is used. It corresponds to IMPLICIT as defined in the Fieldbus Data Link Layer specification.

I.2.1.4.5 DleRemoteConf

This attribute specifies, when true, that the remote DLL immediate Ack shall be used.

I.2.1.5 ExplicitQueue

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

I.2.1.5.1 QueueBindings

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

I.2.1.5.2 MaxQueueDepth

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

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

I.2.1.5.3 MaxDlsduSize

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

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

I.2.2 DLL Services

Refer to annex C for DLL service descriptions.

I.3 QUB-CL ARPM State Machine

I.3.1 QUB-CL ARPM States

The defined states and their descriptions of the QUB-CL ARPM are listed below:

Table I.4 – QUB-CL ARPM States

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.

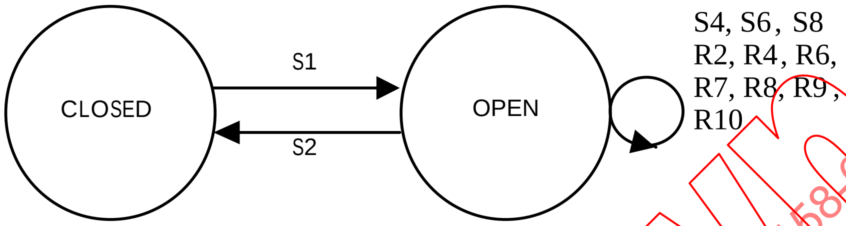


Figure I.1 – State Transition Diagram of the QUB-CL ARPM

I.3.2 QUB-CL ARPM State Table

Table I.5 – QUB-CL ARPM State Table - Sender Transactions

#	Current State	Event Action	Next State

S1	CLOSED	EST_req
		
		EST_cnf(+) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			OPEN

S2	OPEN	Abort_req (no action taken)	CLOSED

S3	OPEN	CS_req
		&& ConfigurationType = "Linked"
		&& Role = "Client" || "Peer"
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		called_address := Remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "CS_ReqPDU",
		fal_data := user_data)
		}
			OPEN

S4	OPEN	CS_req
		&& RemoteAddressConfigurationType = "Free"
		&& Role = "Client" || "Peer"
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		called_address := remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "CS_ReqPDU",
		fal_data := user_data)
		}
			OPEN

#	Current State	Event Action	Next State

S5	OPEN	CS_rsp
		&& ConfigurationType = “Linked”
		&& Role = “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		called_address := Remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "CS_RspPDU",
		fal_data := user_data)
		}
			OPEN

S6	OPEN	CS_rsp
		&& RemoteAddressConfigurationType = “Free”
		&& Role = “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		called_address := remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "CS_RspPDU",
		fal_data := user_data)
		}
			OPEN

S7	OPEN	UCS_req
		&& ConfigurationType = “Linked”
		&& Role = “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		called_address := Remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "UCS_PDU",
		fal_data := user_data)
		}
			OPEN

S8	OPEN	UCS_req
		&& RemoteAddressConfigurationType = “Free”
		&& Role = “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Unitdata_req",
		arep_id := GetArepId (),
		called_address := remote_dlsap_address,
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "UCS_PDU",
		fal_data := user_data)
		}
			OPEN

Table I.6 – QUB-CI ARPM State Table - Receiver Transactions

#	Current State	Event Action	Next State

R1	OPEN	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_Unitdata_ind"
		&& ConfigurationType = "Linked"
		&& RemoteDlsapAddress = calling_address
		&& FAL_Pdu_Type (fal_pdu) = "CS_ReqPDU"
		&& Role = "Server" || "Peer"
		
		CS_ind {
		arep_id := GetArepId (),
		remote_dlsap_address := "null",
		user_data := fal_pdu
		}
			OPEN

R2	OPEN	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_Unitdata_ind"
		&& RemoteAddressConfigurationType = "Free"
		&& FAL_Pdu_Type (fal_pdu) = "CS_ReqPDU"
		&& Role = "Server" || "Peer"
		
		CS_ind {
		arep_id := GetArepId (),
		remote_dlsap_address := calling_address,
		user_data := fal_pdu
		}
			OPEN

R3	OPEN	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_Unitdata_ind"
		&& ConfigurationType = "Linked"
		&& RemoteDlsapAddress = calling_address
		&& FAL_Pdu_Type (fal_pdu) = "CS_RspPDU"
		&& Role = "Client" || "Peer"
		
		CS_cnf {
		arep_id := GetArepId (),
		remote_dlsap_address := "null",
		user_data := fal_pdu
		}
			OPEN

R4	OPEN	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_Unitdata_ind"
		&& RemoteAddressConfigurationType = "Free"
		&& FAL_Pdu_Type (fal_pdu) = "CS_RspPDU"
		&& Role = "Client" || "Peer"
		
		CS_cnf {
		arep_id := GetArepId (),
		remote_dlsap_address := calling_address,
		user_data := fal_pdu
		}
			OPEN

R5	OPEN	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_Unitdata_ind"
		&& ConfigurationType = "Linked"
		&& RemoteDlsapAddress = calling_address
		&& FAL_Pdu_Type (fal_pdu) = "UCS_PDU"
		&& Role = "Server" || "Peer"
		
		UCS_ind {
		arep_id := GetArepId ()
		remote_dlsap_address := "null",
		user_data := fal_pdu
		}
			OPEN

#	Current State	Event Action	Next State

R6	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Unitdata_ind”
		&& RemoteAddressConfigurationType = “Free”
		&& FAL_Pdu_Type (fal_pdu) = “UCS_PDU”
		&& Role = “Client” || “Peer”
		
		UCS_ind {
		arep_id := GetArepId (),
		remote_dlsap_address := calling_address,
		user_data := fal_pdu
		}
			OPEN

R7	OPEN	FAL-PDU_Ind
		&& dmpm_service_name = "DMPM_Unidata_cnf"
		&& FALPdu_Type(fal-pdu) = "CS_Req PDU"
		&& Role = "Client || Peer"
		&& dl-status <> success
		
		CS_Cnf {
		arep_id := GetArepId(),
		remote_dlsap_address := null if Linked otherwise calling_address,
		user_data := null,
		result := dl_status
		}
			OPEN

R8	OPEN	FAL-PDU_Ind && dmpm_service_name = "DMPM_Unidata_cnf" && FALPdu_Type(fal-pdu) = "CS_Req PDU" && Role = "Client || Peer" && dl-status = "success" (no action)	OPEN

R9	OPEN	FAL-PDU_Ind
		&& dmpm_service_name = "DMPM_Unidata_cnf"
		&& FALPdu_Type(fal-pdu) = "UCS_Req PDU"
		&& Role = "Client || Peer"
		
		FSTS_Ind{
		arep_Id := GetArepId()
		reported_status := dl_status
		}
			OPEN

R10	OPEN	FAL-PDU_Ind && dmpm_service_name = "DMPM_Unidata_cnf" && FALPdu_Type(fal-pdu) = "CS_Rsp PDU" && Role = "Server || Peer" (no action)	OPEN

I.3.3 Functions used by QUB-CI ARPM

Table I.7 – Function GetArepId ()

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

Table I.8 – 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 I.9 – Function FAL-Pdu_Type

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

Annex J
(normative)

Queued Usertriggered Bidirectional-Segmentation (QUB-Seg) ARPM

(Normative)

J.1 Primitive Definitions

J.1.1 Primitives Exchanged between ARPM and FSPM

Table J.1 – 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.
UCS_req	FSPM	remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) request primitive from the FSPM to the ARPM.

Table J.2 – 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.
UCS_ind	ARPM	arep_id, remote_dlsap_address, 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.

J.2 Parameters of FSPM/ARPM Primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in table J.3.

Table J.3 – 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 specification.
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.
duplicate_fal_sdu	This parameter conveys value that is used for the Duplicate_FAL-SDU parameter.
remote_dlsap_address	This parameter conveys value that is used for the Remote_DLSAP_Address parameter.
status	This parameter conveys value that is used for the Status parameter.
reported_status	This parameter conveys a Data Link Layer event status.
local_timeliness	This parameter conveys value that is used for the Local_Timeliness parameter.
remote_timeliness	This parameter conveys value that is used for the Remote_Timeliness parameter.

J.2.1 DLL Mapping of QUB-Seg AREP Class

This clause describes the mapping of the QUB-Seg 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 specification; rather, it defines how they are used by this AREP class. A means to configure and monitor the values of these attributes will be provided in the future IEC 61158-7.

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

This class is defined to support the concurrent use of confirmed and unconfirmed services on the same AR as well as the conveyance of services with a maximum data size of 64 kbytes by providing a segmentation functionality, which is invisible to the FAL user. This class uses the DDL features of CLASSICAL connections to ensure the correct segmentation and reassembling of FAL services.

CLASS: QubSeg

PARENT CLASS: QueuedUser-TriggeredBidirectionalSegmentationAREP

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	DlsapRole (Basic)
4	(m)	Attribute:	QosParameterSet
4.1	(m)	Attribute:	DlcepClass (Peer)
4.2	(m)	Attribute:	DlcepDataDeliveryFeatures
4.2.1	(m)	Attribute:	FromRequestorToResponder (Classical)
4.2.2	(m)	Attribute:	FromResponderToRequestor (Classical)
4.3	(m)	Attribute:	Priority
4.3.1	(m)	Attribute:	DllPriority (Urgent, Normal, TimeAvailable)
4.3.2	(m)	Attribute:	DllPriorityNegotiated (Urgent, Normal, TimeAvailable)
4.4	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
4.5	(m)	Attribute:	ResidualActivity
4.5.1	(m)	Attribute:	ResidualActivityAsSender (True, False)
4.5.2	(m)	Attribute:	ResidualActivityAsReceiver (True, False)
4.6	(m)	Attribute:	MaxConfirmDelay
4.6.1	(m)	Attribute:	MaxConfirmDelayOnDlConnect
4.6.2	(m)	Attribute:	MaxConfirmDelayOnDlData
4.7	(m)	Attribute:	DlSchedulingPolicy (Implicit)
4.8	(m)	Attribute:	ExplicitQueue (True, False)
4.9	(c)	Constraint:	ExplicitQueue = True
4.9.1	(m)	Attribute:	MaxDlsduSizes
4.9.1.1	(m)	Attribute:	MaxDlsduSizeFromRequestor
4.9.1.2	(m)	Attribute:	MaxDlsduSizeFromResponder
4.9.1.3	(m)	Attribute:	MaxDlsduSizeFromRequestorNegotiated
4.9.1.4	(m)	Attribute:	MaxDlsduSizeFromResponderNegotiated
4.9.2	(m)	Attribute:	QueueBindings
4.9.2.1	(m)	Attribute:	SendingBufferOrQueueIdentifier
4.9.2.2	(m)	Attribute:	ReceivingBufferOrQueueIdentifier
4.9.3	(m)	Attribute:	MaxQueueDepth
4.9.3.1	(m)	Attribute:	MaxSendingQueueDepth
4.9.3.2	(m)	Attribute:	MaximumReceivingQueueDepth

DLL SERVICES:

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

J.2.1.1 Attributes

J.2.1.1.1 LocalDlcepAddress

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

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

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

J.2.1.1.2 RemoteDlcepAddress

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

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

J.2.1.1.3 DlsapRole

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

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

J.2.1.1.4 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.

J.2.1.1.4.1 DlcepClass

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

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

J.2.1.1.4.2 DlcepDataDeliveryFeatures

These two attributes specify data delivery features of the DLL.

The permitted values Classical and Disordered correspond, respectively to CLASSICAL and DISORDERED defined by the Data Link Layer specification.

The FromRequestorToResponder and FromResponderToRequestor attributes shall have the same value.

J.2.1.1.4.2.1 FromRequestorToResponder

This attribute specifies the DLL data delivery feature of the DLPDUs sent from the AREP whose Initiator attribute has a value of “True” to the remote AREP. It supplies the value for the “DLCEP data delivery features from requestor to responder(s)” parameter defined in the DLL. Only the value ” CLASSICAL ” is allowed since disordered segments would corrupt the FAL PDUs.

J.2.1.1.4.2.2 FromResponderToRequestor

This attribute specifies the DLL data delivery feature of the DLPDUs sent from the AREP whose Initiator attribute has a value of “False” to the remote AREP. It supplies the value for the “DLCEP data delivery features from responder(s) to requestor” parameter defined in the DLL. Only the value ” CLASSICAL ” is allowed since disordered segments would corrupt the FAL PDUs.

J.2.1.1.4.3 Priority

J.2.1.1.4.3.1 DllPriority

This attribute specifies the configured value of the DLL priority.

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

J.2.1.1.4.3.2 DllPriorityNegotiated

This attribute specifies the negotiated value of the DLL priority.

J.2.1.1.4.4 DlpduAuthentication

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

This attribute supplies the value for the “DLPDU-authentication” parameter of the DLL. The permitted value Ordinary, Source, and Maximal correspond to ORDINARY, Source, and MAXIMAL, respectively, as defined in the Fieldbus Data Link Layer specification.

J.2.1.1.4.5 ResidualActivityAsSender

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

J.2.1.1.4.6 ResidualActivityAsReceiver

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

J.2.1.1.4.7 MaxConfirmDelay

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

J.2.1.1.4.7.1 MaxConfirmDelayOnDlConnect

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

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

J.2.1.1.4.7.2 MaxConfirmDelayOnDlData

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

This attribute supplies the value for the “Maximum confirmation delay on DL-Data” parameter specified in the Data Link Layer specification (IEC 61158-3).

J.2.1.1.4.8 DlSchedulingPolicy

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

This attribute supplies the value for the “DL-Scheduling-policy” parameter of the DLL. The permitted value Implicit corresponds to IMPLICIT defined in the Fieldbus Data Link Layer specification.

J.2.1.1.4.9 ExplicitQueue

J.2.1.1.4.9.1 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.

J.2.1.1.4.9.2 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.

J.2.1.1.4.9.3 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.

J.2.1.1.4.9.4 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.

J.2.1.1.4.9.5 SendingBufferOrQueueIdentifier

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

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

J.2.1.1.4.9.6 ReceivingBufferOrQueueIdentifier

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

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

J.2.1.1.4.9.7 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.

J.2.1.1.4.9.8 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.

J.2.1.2 DLL Services

Refer to annex C for DLL service descriptions.

J.3 QUB-Seg ARPM State Machine

J.3.1 QUB-Seg ARPM States

The defined states and their descriptions of the QUB-Seg ARPM are listed below:

Table J.4 – QUB-Seg ARPM States

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.

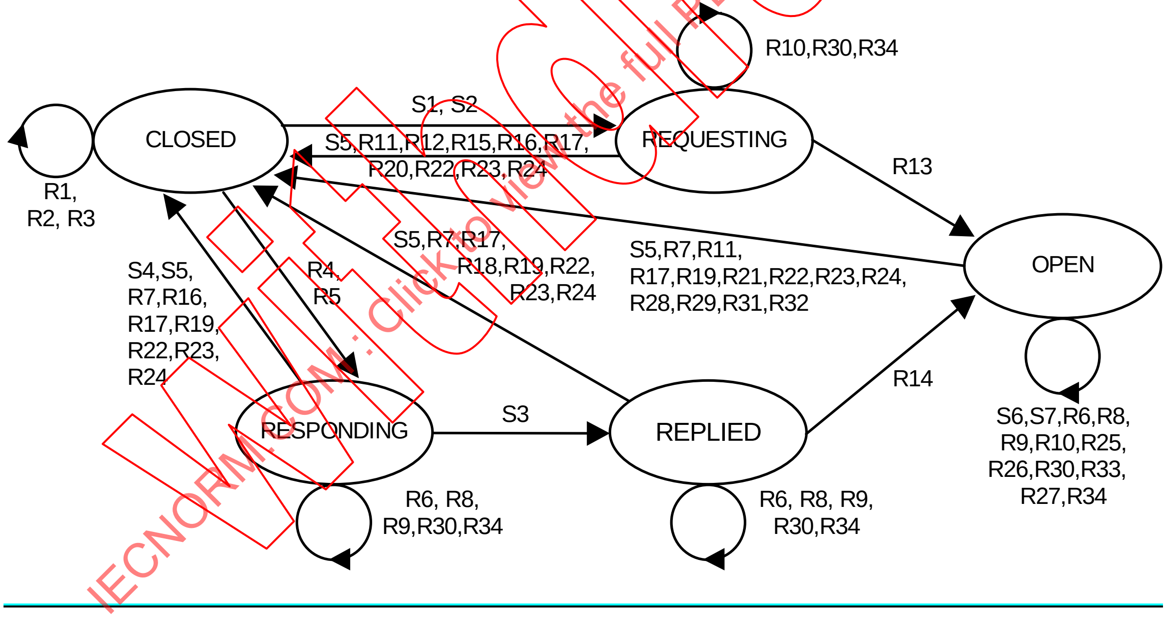


Figure J.1 – State Transition Diagram of QUB-Seg ARPM

J.3.2 QUB-Seg ARPM State Table

Table J.5 – QUB-Seg ARPM State Table - Sender Transactions

#	Current State	Event 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	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

S4	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

#	Current State	Event Action	Next State

S5	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)
		  }
			CLOSED

S6	OPEN	CS_req
		&& Role = “Client” || “Peer”
		&& PDU length < MaxDlsduSizeFromRequestorNegotiated
		
		  FAL-PDU_req {
		    dmpm_service_name := "DMPM_Data_req",
		    arep_id := GetArepId (),
		    dlsdu := BuildFAL-PDU (
		      fal_pdu_name := "CS_ReqPDU",
		      fal_data := user_data)
		  }
			OPEN

S7	OPEN	CS_rsp
		&& Role = “Server” || “Peer”
		&& PDU length < MaxDlsduSizeFromRequestorNegotiated
		
		  FAL-PDU_req {
		    dmpm_service_name := "DMPM_Data_req",
		    arep_id := GetArepId (),
		    dlsdu := BuildFAL-PDU (
		      fal_pdu_name := "CS_RspPDU",
		      fal_data := user_data)
		  }
			OPEN

S8	OPEN	UCS_req
		&& Role = “Server” || “Peer”
		&& PDU length < MaxDlsduSizeFromRequestorNegotiated
		
		  FAL-PDU_req {
		    dmpm_service_name := "DMPM_Data_req",
		    arep_id := GetArepId (),
		    dlsdu := BuildFAL-PDU (
		      fal_pdu_name := "UCS_ReqPDU",
		      fal_data := user_data)
		  }
			OPEN

#	Current State	Event Action	Next State

S9	OPEN	CS_req
		&& Role = “Client” || “Peer”
		&& PDU length >= MaxDlsduSizeFromRequestorNegotiated
		
		for (n := 1 to (PDU length - 1) div (MaxDlsduSizeFromRequestorNegotiated-1)+1)
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-Segment (
		fal_pdu_name := "CS_ReqPDU",
		fal_data := Segment _data, n)
		}
		endfor
		
		NOTE 1 When the length of the FAL-PDU is greater or equal to max size Dlsdu negotiated, the QUB_Seg protocol splits the FAL-PDU into N ordered segment_data numbered from 1 to N. The first Segment_data contain the FAL_Header.
		
		NOTE 2 For each Segment_data, the function “Build-FAL-Segment” builds an APDU-Segment := FAL Header followed with Segment_data without any gap. The header shall have its bit 4 := 1 for the Segment_data 1 to N-1, and 0 for the Segment_data N.
		
		NOTE 3 The first APDU-Segment to be sent through the the network shall contain the Segment_data 1 and so on.
			OPEN

S10	OPEN	CS_rsp
		&& Role = “Server” || “Peer”
		&& PDU length >= MaxDlsduSizeFromRequestorNegotiated
		
		for (n := 1 to (PDU length - 1) div (MaxDlsduSizeFromRequestorNegotiated-1)+1)
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-Segment (
		fal_pdu_name := "CS_RspPDU",
		fal_data := Segment _data, n)
		}
		endfor
		
		NOTE 1 When the length of the FAL-PDU is greater or equal to max size Dlsdu negotiated, the QUB_Seg protocol splits the FAL-PDU into N ordered segment_data numbered from 1 to N. The first Segment_data shall contain the FAL_Header.
		
		NOTE 2 For each segment_data, the function “Build-FAL-Segment” builds an APDU-Segment := FAL Header followed with Segment_data without any gap. The header shall have its bit 4 := 1 for the segment_data 1 to N-1, and 0 for the segment_data N.
		
		NOTE 3 The first APDU-Segment to be sent through the network shall contain the Segment_data 1 and so on.
			OPEN

#	Current State	Event Action	Next State

S11	OPEN	UCS_req
		&& Role = "Server" || "Peer"
		&& PDU length >= MaxDlsduSizeFromRequestorNegotiated
		
		for (n := 1 to (PDU length - 1) div (MaxDlsduSizeFromRequestorNegotiated-1)+1)
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-Segment (
		fal_pdu_name := "UCS_ReqPDU",
		fal_data := SEgment _data, n)
		}
		endfor
		
		NOTE 1 When the length of the FAL-PDU is greater or equal to max size Dlsdu negotiated, the QUB_Seg protocol splits the FAL-PDU into N ordered Segment_data numbered from 1 to N. The first Segment_data shall contain the FAL_Header.
		
		NOTE 2 For each Segment_data, the function "Build-FAL-Segment" builds an APDU-Segment := FAL Header followed with Segment_data without any gap. The header shall have its bit 4 := 1 for the Segment_data 1 to N-1, and 0 for the Segment_data N.
		
		NOTE 3 The first APDU-Segment to be sent through the the network contains the Segment_data 1 and so on.
			OPEN

Table J.6 – ARPM state table - Receiver transactions

#	Current State	Event Action	Next State

R1	CLOSED	Connect_ind
		&& Initiator = "True"
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Disconnect_req",
		arep_id := GetArepId (),
		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 (),
		reason := "Invalid FAL-PDU",
		dlsdu := "null"
		}
			CLOSED

R3	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 (),
		reason := "Remote Address Mismatch",
		dlsdu := "null"
		}
			CLOSED

#	Current State	Event Action	Next State

R4	CLOSED	Connect_ind
		&& Initiator = "False"
		&& FAL_Pdu_Type (dls_user_data) = "EST_ReqPDU"
		&& 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

R5	CLOSED	Connect_ind
		&& Initiator = "False"
		&& FAL_Pdu_Type (dls_user_data) = "EST_ReqPDU"
		&& 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

R6	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 (),
		reason := "Remote Address Mismatch",
		dlsdu := "null"
		}
			SAME

R7	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = "False"
		&& RemoteDlcepAddress = calling_dlcep_address
		
		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

#	Current State	Event Action	Next State

R8	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 (),
		reason := "AREP Busy",
		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 (),
		reason := "Invalid FAL-PDU",
		dlsdu := “null”
		}
			SAME

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

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

#	Current State	Event Action	Next State

R12	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

R13	REQ	Connect_cnf
		&& FAL_Pdu_Type (dls_user_data) = "EST_RspPDU"
		
		ResetIntermediatePDU ()
		
		MaxDlsduSizeFromRequestorNegotiated := dlsdu_size_from_requestor,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		DllPriorityNegotiated := dll_priority,
		
		EST_cnf(+) {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			OPEN

R14	REPL	FAL-PDU_ind && dmpm_service_name = "DMPM_Connection_Established_ind" ResetIntermediatePDU ()	OPEN

R15	REQ	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& FAL_Pdu_Type (fal_pdu) = "EST_ErrPDU"
		
		EST_cnf(-) {
		arep_id := GetArepId (),
		user_data := dls_user_data
		}
			CLOSED

R16	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

#	Current State	Event Action	Next State

R17	NOT CLOSED	FAL-PDU_ind
		&& dmpm_service_name = "DMPM_Disconnect_ind"
		&& fal_pdu <> "null"
		&& 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

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	REPL
	RSP
	OPEN
		FAL-PDU_ind
		&& 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

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

#	Current State	Event Action	Next State

R21	OPEN	FAL-PDU_ind
		&& ((dmpm_service_name <> “DMPM_Disconnect_ind”)
		&& (dmpm_service_name <> “DMPM_Data_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

R22	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"
		}
			CLOSED

R23	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"
		}
			CLOSED

R24	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"
		}
			CLOSED

#	Current State	Event Action	Next State

R25	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Peer” || “Server”
		&& FAL_Pdu_Type (fal_pdu) = “CS_ReqPDU”
		&& AddSegment (fal_pdu) = “OK”
		
		if (MoreFollows(fal_pdu) = “False”)
		  CS_ind {
		    arep_id := GetArepId (),
		    user_data := GetIntermediatePDU ()
		  }
		endif
		
		NOTE When the length of an FAL-PDU is greater or equal to the max size Dlsdu negotiated, the DLL delivers the received APDU segments, starting with the APDU segment 1 and ending with the APDU segment N (the same order as they were sent by the sender). Each APDU_Segment contains an FAL header and a Segment_data. The DLL disconnects the channel if a DLL frame (segment) is missing, the QUB_Seg protocol reassembles the Segment_data of the N received APDU segments. The N-1 APDU_Segments shall have a Header with bit 4 = 1, and the last APDU_Segment shall have a header with bit 4 = 0. The function “AddSegment” removes the header of the APDU_Segment and appends the Segment_data to the previous received Segment_data. Once the N Segment_data are appended together without any gap, the function “GetintermediatePDU” gives the original FAL-APDU.
			OPEN

R26	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Client” || “Peer”
		&& FAL_Pdu_Type (fal_pdu) = “CS_RspPDU”
		&& AddSegment (fal_pdu) = “OK”
		
		if (MoreFollows(fal_pdu) = “False”)
		  CS_cnf {
		    arep_id := GetArepId (),
		    user_data := fal_pdu
		  }
		endif
		
		NOTE When the length of an FAL-PDU is greater or equal to the max size Dlsdu negotiated, the DLL delivers the received APDU segments, starting with the APDU segment 1 and ending with the APDU segment N (the same order as they were sent by the sender). Each APDU_Segment contains a FAL header and a Segment_data. The DLL disconnects the channel if a DLL frame (segment) is missing, the QUB_Seg protocol reassembles the Segment_data of the N received APDU segments. The N-1 APDU_segments shall have a Header with bit 4 = 1, and the last APDU_Segment shall have a header with bit 4 = 0. The function “AddSegment” removes the header of the APDU_Segment and appends the Segment_data to the previous received Segment_data. Once the N Segment_data are appended together without any gap, the function “GetintermediatePDU” gives the original FAL-APDU.
			OPEN

#	Current State	Event Action	Next State

R27	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Server” || “Peer”
		&& FAL_Pdu_Type (fal_pdu) = “UCS_ReqPDU”
		&& AddSegment (fal_pdu) = “OK”
		
		if (MoreFollows(fal_pdu) = “False”)
		UCS_ind {
		arep_id := GetArepId (),
		remote_dlsap_address = “null”,
		user_data := fal_pdu
		}
		endif
		
		NOTE When the length of an FAL-PDU is greater or equal to the max size Dlsdu negotiated, the DLL delivers the received APDU segments, starting with the APDU segment 1 and ending with the APDU segment N (the same order as they were sent by the sender). Each APDU_Segment contains a FAL header and a Segment_data. The DLL disconnects the channel if a DLL frame (segment) is missing, the QUB_Seg protocol reassembles the Segment_data of the N received APDU segments. The N-1 APDU_Segments shall have a Header with bit 4 = 1, and the last APDU_Segment shall have a header with bit 4 = 0. The function “AddSegment” removes the header of the APDU_Segment and appends the Segment_data to the previous received Segment_data. Once the N Segment_data are appended together without any gap, the function “GetintermediatePDU” gives the original FAL-APDU.
			OPEN

R28	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Peer” || “Server”
		&& FAL_Pdu_Type (fal_pdu) = “CS_ReqPDU_|| UCS_ReqPDU_”
		&& AddSegment (fal_pdu) <> “OK”
		
		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"
		}
			CLOSED

R29	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Peer” || “Client”
		&& (FAL_Pdu_Type (fal_pdu) = “CS_RspPDU”)
		&& AddSegment (fal_pdu) <> “OK”
		
		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"
		}
			CLOSED

#	Current State	Event Action	Next State

R30	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_Cnf(-)”
		&& Role = “Server”
		&& FAL_Pdu_Type (fal_pdu) <> “CS_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”,
		additional_detail := "null"
		}
			CLOSED

R31	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_Cnf”
		&& Role = “Client”
		&& FAL_Pdu_Type (fal_pdu) <> “CS_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”,
		additional_detail := "null"
		}
			CLOSED

R32	OPEN	FAL-PDU_ind
		&& Role = “Peer”
		&& dmpm_service_name = “DMPM_Data_ind”
		&& ((FAL_Pdu_Type (fal_pdu) <> “CS_ReqPDU”)
		&& (FAL_Pdu_Type (fal_pdu) <> “CS_RspPDU”)
		&& (FAL_Pdu_Type (fal_pdu) <> “UCS_ReqPDU”))
		
		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"
		}
			CLOSED

#	Current State	Event Action	Next State

R33	OPEN	FAL-PDU_Ind
		&& dmpm-service-name = "DMPM_Data_Cnf(-)"
		&& Role = "Client || Peer"
		&&FAL_Pdu_Type(fal_pdu) = "UCS_ReqPDU"
		
		FAL_PDU_Req{
		dmpm-service-name := "DMPM_Disconnect-req"
		arep-Id := GetArepId(),
		reason := "dl-status",
		dl-sddu := "null"
		}
		
		Abort_Ind{
		arep_Id := GetArepId()
		locally-generated := "true"
		Identifier := "Data Link Layer"
		reason code := reason
		additional_detail := "null"
		}
			OPEN

R34	NOT CLOSED	ErrorToARPM (no actions taken) NOTE 1 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. NOTE 2 This event mainly occurs when the DLL has not been able to convey a DL service to the remote partner within the specified time interval. In this case an implementation may also decide to pass negative responses for all confirmed FAL service requests that are affected by this failure to the respective FAL user.	SAME

J.3.3 Functions used by QUB-Seg ARPM

Table J.7 – Function GetArepId

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

Table J.8 – 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 J.9 – 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 J.10 – 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 J.11 – 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 J.12 – 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.			

Table J.13 – Function BuildFAL-Segment

Name	BuildFAL-Segment	Used in	ARPM
Input		Output	
fal_pdu_name, fal_data, segment_number		dlsdu	
Function			
Builds an FAL-Segment out of the parameters given as input variables. The segment_number must be >= 1 and defines, which part of the fal_pdu is to be put in the returned segment. This function adds the FAL-Header to each segment. It sets bit 4 of the FAL-Header to 1, indicating that there are more segments to be sent, when it is not the last segment. Otherwise bit 4 is set to 0. Except for this bit the FAL-Header is identical in all segments of one FAL-PDU.			

Table J.14 – Function MoreFollows

Name	MoreFollows	Used in	ARPM
Input		Output	
dls_user_data		Boolean value	
Function			
This function inspects the FAL-PDU that is conveyed in the dls_user_data parameter and returns "True", if bit5 of the FAL-Header is set. Otherwise it returns "False".			

NOTE The following three functions make use of a persistent variable IntermediatePDU, in which all segments of the current FAL User Data are stored by the receiver of this data.

Table J.15 – Function ResetIntermediatePDU

Name	ResetIntermediatePDU	Used in	ARPM
Input		Output	
(none)		(none)	
Function			
This function removes all segments which have so far been collected from the IntermediatePDU.			

Table J.16 – Function AddSegment

Name	AddSegment	Used in	ARPM
Input		Output	
dls_user_data		Error Code	
Function			
This function adds the received dls_user_data as a new segment to the IntermediatePDU. It checks segments reassembling protocol error. If no error was detected during these checks, the error code "OK" is returned.			

Table J.17 – Function GetIntermediatePDU

Name	GetIntermediatePDU	Used in	ARPM
Input		Output	
(none)		fal_user_data	
Function			
This function returns the complete FAL user data, which was received in multiple segments. After this function call the variable IntermediatePDU does not contain any segments.			

Annex K
(normative)

Queued Usertriggered Bidirectional-Flow Control (QUB-FC) ARPM

K.1 QUB-FCPrimitive Definitions

K.1.1 Primitives Exchanged between ARPM and FSPM

Table K.1 – 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.
AR_XON_OFF_req	FSPM	remote_dlsap_address, user_data	This is an FAL internal primitive used to convey AR-XON-OFF request primitive from the FSPM to the ARPM.

Table K.2 – 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, duplicate_fal_sdu, user_data	This is an FAL internal primitive used to convey an Unconfirmed Acknowledged Send (UCA) indication primitive from the ARPM to the FSPM.
AR_XON_OFF_ ind	ARPM	arep_id, remote_dlsap_address, user_data	This is an FAL internal primitive used to convey a AR-XON-OFF indication primitive from the ARPM to the FSPM.

K.2 Parameters of FSPM/ARPM Primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in table K.3.

Table K.3 – 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 specification.
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.
Duplicate_fal_sdu	This parameter conveys value that is used for the Duplicate_FAL-SDU parameter.
Remote_dlsap_address	This parameter conveys value that is used for the Remote_DLSAP_Address parameter.
Status	This parameter conveys value that is used for the Status parameter.
Reported_status	This parameter conveys a Data Link Layer event status.
Local_timeliness	This parameter conveys value that is used for the Local_Timeliness parameter.
Remote_timeliness	This parameter conveys value that is used for the Remote_Timeliness parameter.

K.2.1 DLL Mapping of QUB-FC AREP Class

This clause describes the mapping of the QUB-FC 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 specification; rather, it defines how they are used by this AR class. A means to configure and monitor the values of these attributes will be provided in the future IEC 61158-7.

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

CLASS: QubFC

PARENT CLASS: QueuedUser-TriggeredBidirectional-FlowControlAREP

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	DlsapRole (Basic)
4	(m)	Attribute:	QosParameterSet
4.1	(m)	Attribute:	DlcepClass (Peer)
4.2	(m)	Attribute:	DlcepDataDeliveryFeatures
4.2.1	(m)	Attribute:	FromRequestorToResponder (Classical, Disordered)
4.2.2	(m)	Attribute:	FromResponderToRequestor (Classical, Disordered)
4.3	(m)	Attribute:	Priority
4.3.1	(m)	Attribute:	DllPriority (Urgent, Normal, TimeAvailable)
4.3.2	(m)	Attribute:	DllPriorityNegotiated (Urgent, Normal, TimeAvailable)
4.4	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
4.5	(m)	Attribute:	ResidualActivity
4.5.1	(m)	Attribute:	ResidualActivityAsSender (True, False)
4.5.2	(m)	Attribute:	ResidualActivityAsReceiver (True, False)
4.6	(m)	Attribute:	MaxConfirmDelay
4.6.1	(m)	Attribute:	MaxConfirmDelayOnDlConnect
4.6.2	(m)	Attribute:	MaxConfirmDelayOnDlData
4.7	(m)	Attribute:	DlSchedulingPolicy (Implicit)
4.8	(m)	Attribute:	ExplicitQueue (True, False)
4.9	(c)	Constraint:	ExplicitQueue = True
4.9.1	(m)	Attribute:	MaxDlsduSizes
4.9.1.1	(m)	Attribute:	MaxDlsduSizeFromRequestor
4.9.1.2	(m)	Attribute:	MaxDlsduSizeFromResponder
4.9.1.3	(m)	Attribute:	MaxDlsduSizeFromRequestorNegotiated
4.9.1.4	(m)	Attribute:	MaxDlsduSizeFromResponderNegotiated
4.9.2	(m)	Attribute:	MaxQueueDepth
4.9.2.1	(m)	Attribute:	MaxSendingQueueDepth
4.9.2.2	(m)	Attribute:	MaximimReceivingQueueDepth

DLL SERVICES:

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

K.2.1.1 Attributes

K.2.1.1.1 LocalDlcepAddress

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

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

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

K.2.1.1.2 RemoteDlcepAddress

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

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

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

K.2.1.1.3 DlsapRole

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

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

K.2.1.1.4 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.

K.2.1.1.4.1 DlcepClass

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

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

K.2.1.1.4.2 DlcepDataDeliveryFeatures

These two attributes specify data delivery features of the DLL.

The permitted values Classical and Disordered correspond, respectively to CLASSICAL and DISORDERED defined by the Data Link Layer specification.

The FromRequestorToResponder and FromResponderToRequestor attributes shall have the same value.

K.2.1.1.4.2.1 FromRequestorToResponder

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

K.2.1.1.4.2.2 FromResponderToRequestor

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

K.2.1.1.5 Priority

K.2.1.1.5.1 DllPriority

This attribute specifies the configured value of the DLL priority.

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

K.2.1.1.5.2 DllPriorityNegotiated

This attribute specifies the negotiated value of the DLL priority.

K.2.1.1.6 DlpduAuthentication

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

This attribute supplies the value for the “DLPDU-authentication” parameter of the DLL. The permitted value Ordinary, Source, and Maximal correspond to ORDINARY, Source, and MAXIMAL, respectively, as defined in the Fieldbus Data Link Layer specification.

K.2.1.1.6.1 ResidualActivity

K.2.1.1.6.1.1 ResidualActivityAsSender

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

K.2.1.1.6.1.2 ResidualActivityAsReceiver

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

K.2.1.1.6.2 MaxConfirmDelay

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

K.2.1.1.6.2.1 MaxConfirmDelayOnDlConnect

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

This attribute supplies the value for the "Maximum confirmation delay on DL-Connect, DL-Reset and DL-Subscriber-Query" parameter specified in the Data Link Layer specification.

K.2.1.1.6.2.2 MaxConfirmDelayOnDlData

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

This attribute supplies the value for the "Maximum confirmation delay on DL-Data" parameter specified in the Data Link Layer specification.

K.2.1.1.6.3 DlSchedulingPolicy

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

This attribute supplies the value for the "DL-Scheduling-policy" parameter of the DLL. The permitted value Implicit corresponds to IMPLICIT defined in the Data Link Layer specification.

K.2.1.1.6.4 ExplicitQueue

K.2.1.1.6.4.1 MaxDlsduSize

K.2.1.1.6.4.1.1 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.

K.2.1.1.6.4.1.2 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.

K.2.1.1.6.4.1.3 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.

K.2.1.1.6.4.1.4 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.

K.2.1.1.6.4.2 MaxQueueDepth

K.2.1.1.6.4.2.1 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.

K.2.1.1.6.4.2.2 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.

K.2.2 DLL Services

Refer to annex C for DLL service descriptions.

K.3 QUB-FC ARPM State Machine

K.3.1 QUB-FC ARPM States

The defined states and their descriptions of the QUB-FC ARPM are listed below:

Table K.4 – QUB-FC ARPM States

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.

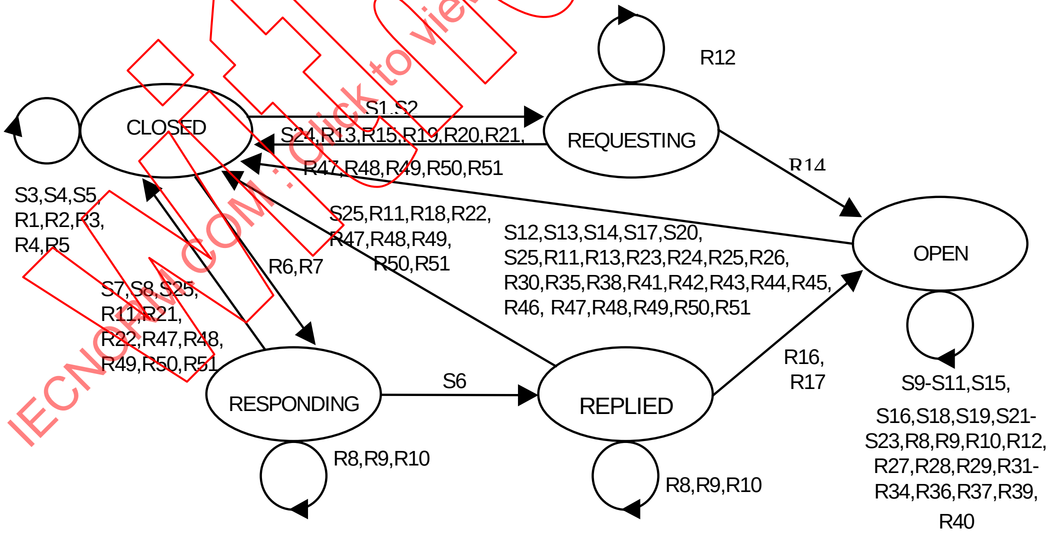


Figure K.1 – State Transition Diagram of QUB-FC ARPM

K.3.2 QUB-FC ARPM State Table

Table K.5 – QUB-FC ARPM State Table - Sender Transactions

#	Current State	Event 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

#	Current State	Event Action	Next State

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

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”
		&& Xon=“True“
		&& 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

#	Current State	Event Action	Next State

S10	OPEN	CS_req
		&& Role = “Client” || “Peer”
		&& Xon=“True“
		&& 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”
		&& Xon=“False“
		
		CS_cnf (-) {
		arep_id := GetArepId (),
		user_data := "null"
		}
			OPEN

S12	OPEN	CS_req
		&& Role = “Client” || “Peer”
		&& Xon=“True“
		&& 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

#	Current State	Event Action	Next State

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

S16	OPEN	AR_XON_OFF_req
		&& Role = “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name := "DMPM_Data_req",
		arep_id := GetArepId (),
		dlsdu := BuildFAL-PDU (
		fal_pdu_name := "AR_XON_OFF_ReqPDU",
		fal_data := user_data)
		}
			OPEN

#	Current State	Event 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
		&& Xon = "True“
		&& 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
		&& Xon = "True“
		&& 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

#	Current State	Event Action	Next State

S20	OPEN	UCA_req
		&& Xon = "True"
		&& 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

S21	OPEN	UCA_req && Xon = "False" (no actions taken)	OPEN

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

#	Current State	Event Action	Next State

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 K.6 – QUB-FC ARPM State Table - Receiver Transactions

#	Current State	Event 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

#	Current State	Event Action	Next State

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

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

#	Current State	Event Action	Next State

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

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

#	Current State	Event Action	Next State

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 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

#	Current State	Event Action	Next State
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
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

#	Current State	Event Action	Next State

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

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

#	Current State	Event Action	Next State
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

R29	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Data_ind”
		&& Role = “Client” || “Peer”
		&& FAL_Pdu_Type (dls_user_data) = “AR_XON_OFF_ReqPDU”
		
		SetXon (user_data),
		
		AR_XON_OFF_ind {
		arep_id := GetArepId (),
		user_data := fal_pdu
		}
			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 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

#	Current State	Event Action	Next State

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

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

#	Current State	Event Action	Next State

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
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

#	Current State	Event Action	Next State

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 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

#	Current State	Event Action	Next State

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

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

#	Current State	Event Action	Next State
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
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

#	Current State	Event Action	Next State

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

K.3.3 Functions used by QUB-FC ARPM

Table K.7 – Function GetArepId ()

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

Table K.8 – Function BuildFAL-PDU

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

Table K.9 – Function FAL_Pdu_Type

Name	FAL_Pdu_Type	Used in	ARPM
Input dls_user_data		Output 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 K.10 – 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 matrix:

	Local Context					
Remote Context	Type	maxOSCC	maxOSCS	maxUCSC	maxUCSS	CI
Type	=					
maxOSCC			≥			
mmaxOSCS		≤				
maxUCSC					≥	
maxUCSS				≤		
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 K.11 – 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 K.12 – 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 K.13 – 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 K.14 – Function StartTimer

Name	StartTimer	Used in	ARPM
Input		Output	
identifier			
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 K.15 – 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 K.16 – Function ResetCounters

Name	ResetCounters	Used in	ARPM
Input		Output	
Function			

This function sets OSCC, UCC, OSCS to zero.

Table K.17 – Function IncrementCounter

Name	IncrementCounter	Used in	ARPM
Input		Output	
identifier			
Function			

This function increments the selected counter.

Table K.18 – Function DecrementCounter

Name	DecrementCounter	Used in	ARPM
Input		Output	
identifier			
Function			

This function decrements the selected counter.

Table K.19 – Function GetCounterValue

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

NOTE The following function makes use of local persistent variable XON_OFF_Flag that stores the current value of flow control from the AR_XON_OFF_ReqPDU. The initial value of XON_OFF_Flag is True.

Table K.20 – Function Xon

Name	Xon	Used in	ARPM
Input		Output	
(non)		Boolean (True/False)	
Function			
This function returns True if XON_OFF_Flag was True otherwise it returns False.			

Table K.21 – Function SetXonOff

Name	SetXonOff	Used in	ARPM
Input		Output	
user_data			
Function			
This function extracts the XON_OFF octet from user_data. It sets XON_XOFF_Flag True if XON_OFF is 'ON' within the PDU otherwise it sets the flag False.			

Annex L
(normative)

Buffered Usertriggered Bidirectional (BUB) ARPM

L.1 Primitive Definitions

L.1.1 Primitives Exchanged between ARPM and FSPM

Table L.1 – 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.
UCS_req	FSPM	remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) request primitive from the FSPM to the ARPM.
FCMP_req	FSPM	(none)	This is an FAL internal primitive used to convey a FAL-Compel (FCMP) request primitive from the FSPM to the ARPM.
GBM_req	FSPM	(none)	This is an FAL internal primitive used to convey a Get-Buffered-Message (GBM) request primitive from the FSPM to the ARPM.

Table L.2 – 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.
UCS_ind	ARPM	arep_id, remote_dlsap_address, 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.
FCMP_cnf	ARPM	arep_id, status	This is an FAL internal primitive used to convey a FAL-Compel (FCMP) confirmation primitive from the ARPM to the FSPM.
GBM_cnf(+)	ARPM	arep_id, duplicate_fal_sdu, user_data, local_timeliness, remote_timeliness	This is an FAL internal primitive used to convey a Get-Buffered-Message (GBM) positive confirmation primitive from the ARPM to the FSPM.
GBM_cnf(-)	ARPM	arep_id	This is an FAL internal primitive used to convey a Get-Buffered-Message (GBM) negative confirmation primitive from the ARPM to the FSPM.
FSTS_ind	ARPM	arep_id, reported_status	This is an FAL internal primitive used to convey a FAL-Status (FSTS) indication primitive from the ARPM to the FSPM.

L.1.1.1 Parameters of FSPM/ARPM Primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in table L.3.

Table L.3 – 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 specification.
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.
duplicate_fal_sdu	This parameter conveys value that is used for the Duplicate_FAL-SDU parameter.
remote_dlsap_address	This parameter conveys value that is used for the Remote_DLSAP_Address parameter.
status	This parameter conveys value that is used for the Status parameter.
reported_status	This parameter conveys a Data Link Layer event status.
local_timeliness	This parameter conveys value that is used for the Local_Timeliness parameter.
remote_timeliness	This parameter conveys value that is used for the Remote_Timeliness parameter.

L.2 DLL Mapping of BUB AREP Class

This clause describes the mapping of the BUB 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 specification; rather, it defines how they are used by this AR class. A means to configure and monitor the values of these attributes will be provided in the future IC 61158-7.

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

CLASS: BubCo

PARENT CLASS: **BufferedUser-TriggeredBidirectionalAREP**

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	DlsapRole (Initiator, Constrained)
4	(m)	Attribute:	QosParameterSet
4.1	(m)	Attribute:	DlcepClass (Peer)
4.2	(m)	Attribute:	DlcepDataDeliveryFeatures
4.2.1	(m)	Attribute:	FromRequestorToResponder (Ordered, Unordered)
4.2.2	(m)	Attribute:	FromResponderToRequestor (Ordered, Unordered)
4.3	(m)	Attribute:	Priority
4.3.1	(m)	Attribute:	DllPriority (Urgent, Normal, TimeAvailable)
4.3.2	(m)	Attribute:	DllPriorityNegotiated (Urgent, Normal, TimeAvailable)
4.4	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
4.5	(m)	Attribute:	ResidualActivity
4.5.1	(m)	Attribute:	ResidualActivityAsSender (True, False)
4.5.2	(m)	Attribute:	ResidualActivityAsReceiver (True, False)
4.6	(m)	Attribute:	MaxConfirmDelay
4.7	(c)	Constraint:	DlsapRole = Initiator
4.7.1	(m)	Attribute:	MaxConfirmDelayOnDlConnect
4.7.2	(m)	Attribute:	MaxConfirmDelayOnDlData
4.8	(m)	Attribute:	DlSchedulingPolicy (Explicit)
4.9	(m)	Attribute:	MaxDlsduSizes
4.9.1	(m)	Attribute:	MaxDlsduSizeFromRequestor
4.9.2	(m)	Attribute:	MaxDlsduSizeFromResponder
4.9.3	(m)	Attribute:	MaxDlsduSizeFromRequestorNegotiated
4.9.4	(m)	Attribute:	MaxDlsduSizeFromResponderNegotiated

DLL SERVICES:

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

L.2.1 Attributes

L.2.1.1 LocalDlcepAddress

This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the "DLCEP-address" parameter of the DLL.

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

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

L.2.1.2 RemoteDlcepAddress

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

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

NOTE 1 The value of this attribute is also carried in the header part of the Establish Request PDU.

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

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

L.2.1.3 DlsapRole

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

This attribute supplies the value for the “DL(SAP)-role” parameter. The permitted values Initiator, Constrained Responder and Unconstrained Responder, correspond respectively to INITIATOR, CONSTRAINED, and UNCONSTRAINED as defined in the Data Link Layer specification.

L.2.1.4 QosParameterSet

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

L.2.1.5 DlcepClass

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

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

L.2.1.6 DlcepDataDeliveryFeatures

These two attributes specify the data delivery features of the DLL.

The permitted values Ordered and Unordered correspond respectively to ORDERED and UNORDERED defined by the Data Link Layer specification.

The FromRequestorToResponder and FromResponderToRequestor attributes shall have the same value.

L.2.1.7 FromRequestorToResponder

This attribute specifies the DLL data delivery feature of the AREP as a sender. It supplies the value for the “DLCEP data delivery features from requestor to responder(s)” parameter defined in the DLL.

L.2.1.8 FromResponderToRequestor

This attribute specifies the DLL data delivery feature of the AREP as a receiver. It supplies the value for the “DLCEP data delivery features from responder(s) to requestor” parameter defined in the DLL.

L.2.1.9 Priority

L.2.1.10 DllPriority

This attribute specifies the DLL priority before negotiation.

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

L.2.1.11 DllPriorityNegotiated

This attribute specifies the DLL priority after negotiation.

L.2.1.12 DlpduAuthentication

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

This attribute supplies the value for the “DLPDU-authentication” parameter of the DLL. The permitted value Ordinary, Source and Maximal correspond to ORDINARY, SOURCE and MAXIMAL respectively, as defined in the Fieldbus Data Link Layer specification.

L.2.1.13 ResidualActivityAsSender

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

L.2.1.14 ResidualActivityAsReceiver

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

L.2.1.15 MaxConfirmDelay

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

L.2.1.16 MaxConfirmDelayOnDlConnect

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

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

L.2.1.17 MaxConfirmDelayOnDlData

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

This attribute supplies the value for the “Maximum confirmation delay on DL-Data” parameter specified in the Fieldbus Foundation Data Link Layer specification.

L.2.1.18 DlSchedulingPolicy

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

This attribute supplies the value for the “DL-Scheduling-policy” parameter of the DLL. The permitted value Explicit corresponds to EXPLICIT defined in the Fieldbus Data Link Layer specification.

L.2.1.19 MaxDlsduSizeFromRequestor

This attribute specifies the maximum length of an FAL-PDU that can be sent from this AREP before negotiation.

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

L.2.1.20 MaxDlsduSizeFromResponder

This attribute specifies the maximum length of an FAL-PDU that can be received by this AREP before negotiation.

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

L.2.1.21 MaxDlsduSizeFromRequestorNegotiated

This attribute specifies the maximum length of an FAL-PDU that can be sent from this AREP after negotiation.

L.2.1.22 MaxDlsduSizeFromResponderNegotiated

This attribute specifies the maximum length of an FAL-PDU that can be received by this AREP after negotiation.

L.3 BUB ARPM State Machine

L.3.1 BUB ARPM States

Table L.4 - BUB ARPM States

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 a “connection established” indication from the DLL.
SAME	Indicates that the next state is the same as the current state.

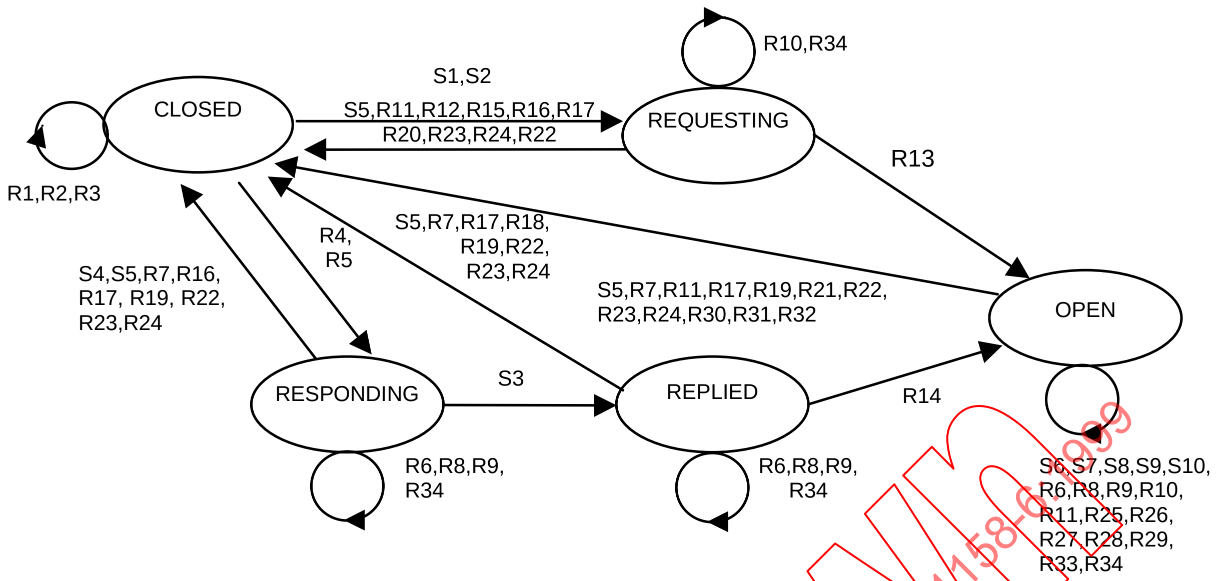


Figure L.1 – State Transition Diagram of BUB ARPM

L.3.2 BUB ARPM State Table

Table L.5 - BUB ARPM State Table - Sender Transactions

#	Current State	Event 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

#	Current State	Event Action	Next State

S3	RSP	EST_rsp(+)
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Connect_rsp”,
		arep_id:= GetArepId(),
		responding_address:= “default dlsap address”,
		local_dlcep_address:= LocalDlcep,
		dlcep_dl_id:= DlcepDlIdentifier,
		dlsdu:= BuildFAL-PDU (
		fal_pdu_name:= “EST_RspPDU”,
		fal_data:= user_data)
		}
			REPL

S4	RSP	EST_rsp(-)
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Disconnect_req”,
		arep_id:= GetArepId(),
		dlcep_dl_id:= DlcepDlIdentifier,
		reason:= “connection rejection-transient condition”,
		dlsdu:= BuildFAL-PDU (
		fal_pdu_name:= “EST_ErrPDU”,
		fal_data:= user_data)
		}
			CLOSED

S5	NOT CLOSED	Abort_req
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Disconnect_req”,
		arep_id:= GetArepId(),
		dlcep_dl_id:= DlcepDlIdentifier,
		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)
		}
			CLOSED

S6	OPEN	CS_req
		&& Role = “Client” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Put_req”,
		arep_id:= GetArepId(),
		dlsdu:= BuildFAL-PDU (
		fal_pdu_name:= “CS_ReqPDU”,
		fal_data:= user_data)
		}
			OPEN

S7	OPEN	CS_rsp
		&& Role = “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Put_req”,
		arep_id:= GetArepId(),
		dlsdu:= BuildFAL-PDU (
		fal_pdu_name:= “CS_RspPDU”,
		fal_data:= user_data)
		}
			OPEN

S8	OPEN	FCMP_req
		&& Role = “Client” || “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Compel_Service_req”,
		arep_id:= GetArepId(),
		action_class:= “Local”
		}
			OPEN

#	Current State	Event Action	Next State

S9	OPEN	GBM_req
		&& Role = “Client” || “Server” || “Peer”
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Get_req”,
		arep_id:= GetArepId()
		}
			OPEN

S10	OPEN	UCS_req
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Put_req”,
		arep_id:= GetArepId(),
		dlsdu:= BuildFAL-PDU (
		fal_pdu_name:= “UCS_ReqPDU”,
		fal_data:= user_data)
		}
			OPEN

Table L.6 - BUB state table - Receiver transactions

#	Current State	Event 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”
		&& 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

#	Current State	Event Action	Next State

R4	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_type (dls_user_data) = “EST_ReqPDU”
		&& RemoteAddressConfigurationType = “Free”
		
		RemoteDlcepAddress:= calling_dlcep_address,
		DlcepDlIdentifier:= dlcep_dl_id,
		MaxDlsduSizeFromRequestorNegotiated := dlsdu_size_from_requestor,
		MaxDlsduSizeFromResponderNegotiated: = dlsdu_size_from_responder,
		
		EST_ind {
		arep_id:= GetArepId (),
		user_data:= dls_user_data
		}
			RSP

R5	CLOSED	Connect_ind
		&& Initiator = “False”
		&& FAL_Pdu_type (dls_user_data) = “EST_ReqPDU”
		&& RemoteAddressConfigurationType = “Linked”
		&& RemoteDlcepAddress = calling_dlcep_address
		
		DlcepDlIdentifier:= dlcep_dl_id,
		MaxDlsduSizeFromRequestorNegotiated := dlsdu_size_from_requestor,
		MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder,
		
		EST_ind {
		arep_id:= GetArepId (),
		user_data:= dls_user_data
		}
			RSP

R6	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

R7	RSP
	REPL
	OPEN
		Connect_ind
		&& Initiator = “False”
		&& 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 Action	Next State

R8	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

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:= “Invalid FAL-PDU”,
		dlsdu:= “null”
		}
			SAME

R10	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

R11	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

#	Current State	Event Action	Next State

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

R13	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

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

R15	REQ	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_Disconnect_ind”
		&& FAL_Pdu_Type (fal_pdu) = “EST-ErrPDU”
		
		EST_cnf(-) {
		arep_id:= GetArepId (),
		user_data:= dls_user_data
		}
			CLOSED

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

#	Current State	Event Action	Next State

R17	NOT CLOSED	FAL-PDU_ind
		&& dmpm_service_name <> “DMPM_Disconnect_ind”
		&& fal_pdu <> “null”
		&& 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

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(),
		dlcep_dl_id:= DlcepDlIdentifier,
		reason:= “Invalid Dl PDU”,
		dlsdu:= “null”
		},
		
		Abort_ind {
		arep_id:= GetArepId (),
		locally_generated:= “True”,
		identifier:= “FAL”,
		reason_code:= “Invalid Dl PDU”
		additional_detail:= “null”
		}
			CLOSED

R19	REPL
	RSP
	OPEN
		FAL-PDU_ind
		&& 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

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

#	Current State	Event Action	Next State

R21	OPEN	FAL-PDU_ind
		&& ((FAL_Pdu_Type (fal_pdu) <> "DMPM_Disconnect_ind")
		&& (FAL_Pdu_Type (fal_pdu) <> "DMPM_Put_Out")
		
		FAL-PDU_req {
		dmpm_service_name:= "DMPM_Disconnect_req",
		arep_id:= GetArepId(),
		dlcep_dl_id:= DlcepDlIdentifier,
		reason:= "Invalid Dl PDU",
		dlsdu:= "null"
		},
		
		Abort_ind {
		arep_id:= GetArepId (),
		locally_generated:= "True",
		identifier:= "FAL",
		reason_code:= "Invalid Dl PDU"
		additional_detail:= "null"
		}
			CLOSED

R22	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"
		}
			CLOSED

R23	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"
		}
			CLOSED

R24	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"
		}
			CLOSED

R25	OPEN	FAL-PDU_ind
		&& Role = "Peer" || "Server"
		&& dmpm_service_name = "DMPM_Buffer_Received_ind"
		&& FAL_Pdu_Type (fal_pdu) = "CS_ReqPDU"
		
		CS_ind {
		arep_id:= GetArepId (),
		user_data:= fal_pdu
		}
			OPEN

#	Current State	Event Action	Next State
R26	OPEN	FAL-PDU_ind && Role = “Client” || “Peer” && dmpm_service_name = “DMPM_ Buffer_Received_ind” && FAL_Pdu_Type (fal_pdu) = “CS_RspPDU” CS_cnf { Arep_id:= GetArepId (), user_data:= fal_pdu }	OPEN
R27	OPEN	FAL-PDU_ind && Role = “Peer”||”Client” || “Server” && dmpm_service_name = “DMPM_Buffer_Sent_ind” FSTS_ind { arep_id:= GetArepId(), reported_status:=“Buffer_Sent” }	OPEN
R28	OPEN	FAL-PDU_ind && Role = “Peer”||”Client” || “Server” && dmpm_service_name = “DMPM_Get_cnf” && status = “success” GMB_cnf(+) { arep_id:= GetArepId(), duplicate_fal_sdu=duplicate_dlsdu, user_data:=fal_pdu }	OPEN
R29	OPEN	FAL-PDU_ind && Role = “Peer”||”Client” || “Server” && dmpm_service_name = “DMPM_Get_cnf” && status <> “success” GMB_cnf(-) { arep_id:= GetArepId() }	OPEN
R30	OPEN	FAL-PDU_ind && Role = “Server” && dmpm_service_name = “DMPM_ Buffer_Received_ind” && FAL_Pdu_Type (fal_pdu) <> “CS_ReqPDU” && FAL_Pdu_Type (fal_pdu) <> "UCS_ReqPDU" FAL-PDU_req { dmpm_service_name:= “DMPM_Disconnect_req”, arep_id:= GetArepId(), dlcep_dl_id:= DlcepDlIdentifier, reason:= “Invalid FAL-PDU”, dlsdu:= “null” }, Abort_ind { arep_id:= GetArepId (), locally_generated:= “True”, identifier:= “FAL”, reason_code:= “Invalid FAL-PDU” additional_detail:= “null” }	CLOSED

#	Current State	Event Action	Next State

R31	OPEN	FAL-PDU_ind
		&& Role = “Client”
		&& dmpm_service_name = “DMPM_ Buffer_Received_ind”
		&& FAL_Pdu_Type (fal_pdu) <> “CS_RspPDU”
		&& FAL_Pdu_Type (fal_pdu) <> "UCS_ReqPDU"
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Disconnect_req”,
		arep_id:= GetArepId(),
		dlcep_dl_id:= DlcepDlIdentifier,
		reason:= “Invalid FAL-PDU”,
		dlsdu:= “null”
		},
		
		Abort_ind {
		arep_id:= GetArepId (),
		locally_generated:= “True”,
		identifier:= “FAL”,
		reason_code:= “Invalid FAL-PDU”
		additional_detail:= “null”
		}
			CLOSED

R32	OPEN	FAL-PDU_ind
		&& Role = “Peer”
		&& dmpm_service_name = “DMPM_Buffer_Received_ind”
		&& ((FAL_Pdu_Type (fal_pdu) <> “CS_ReqPDU”)
		&& (FAL_Pdu_Type (fal_pdu) <> “CS_RspPDU”)
		&& (FAL_Pdu_Type (fal_pdu) <> "UCS_ReqPDU")
		
		FAL-PDU_req {
		dmpm_service_name:= “DMPM_Disconnect_req”,
		arep_id:= GetArepId(),
		dlcep_dl_id:= DlcepDlIdentifier,
		reason:= “Invalid FAL-PDU”,
		dlsdu:= “null”
		},
		
		Abort_ind {
		arep_id:= GetArepId (),
		locally_generated:= “True”,
		identifier:= “FAL”,
		reason_code:= “Invalid FAL-PDU”
		additional_detail:= “null”
		}
			CLOSED

R33	OPEN	FAL-PDU_ind
		&& dmpm_service_name = “DMPM_ Buffer_Received_ind”
		&& FAL_Pdu_Type (fal_pdu) = “UCS_ReqPDU”
		
		UCS_ind {
		arep_id:= GetArepId (),
		user_data:= fal_pdu
		}
			OPEN

R34	NOT CLOSED	ErrorToARPM (no action 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

L.3.3 Functions used by BUB ARPM

Table L.7 – Function GetArepId ()

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

Table L.8 – 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 L.9 – 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 L.10 – 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 L.11 – 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 L.12 – 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.			

Annex M
(normative)

Buffered Networkscheduled Bidirectional (BNB) ARPM

M.1 Primitive Definitions

M.1.1 Primitives Exchanged between ARPM and FSPM

Table M.1 – 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.
UCS_req	FSPM	remote_dlsap_address, user_data	This is an FAL internal primitive used to convey an Unconfirmed Send (UCS) request primitive from the FSPM to the ARPM.

Table M.2 – 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.
UCS_ind	ARPM	arep_id, remote_dlsap_address, 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.

M.1.2 Primitives issued by ARPM to FSPMParameters of FSPM/ARPM Primitives

The parameters used with the primitives exchanged between the FSPM and the ARPM are described in table M.3.

Table M.3 – 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 specification.
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.
duplicate_fal_sdu	This parameter conveys value that is used for the Duplicate_FAL-SDU parameter.
remote_dlsap_address	This parameter conveys value that is used for the Remote_DLSAP_Address parameter.
status	This parameter conveys value that is used for the Status parameter.
reported_status	This parameter conveys a Data Link Layer event status.
local_timeliness	This parameter conveys value that is used for the Local_Timeliness parameter.
remote_timeliness	This parameter conveys value that is used for the Remote_Timeliness parameter.

M.2 DLL Mapping of BNB AREP Class

This clause describes the mapping of the BNB 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 specification; rather, it defines how they are used by this AR class. A means to configure and monitor the values of these attributes will be provided in future IEC 61158-7.

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

CLASS: Bnb

PARENT CLASS: BufferedNetwork-TriggeredBidirectionalAREP

ATTRIBUTES:

1	(m)	KeyAttribute:	LocalDlcepAddress
2	(m)	Attribute:	RemoteDlcepAddress
3	(m)	Attribute:	DlsapRole (Basic)
4	(m)	Attribute:	QosParameterSet
4.1	(m)	Attribute:	DlcepClass (Peer)
4.2	(m)	Attribute:	DlcepDataDeliveryFeatures
4.2.1	(m)	Attribute:	FromRequestorToResponder (Ordered)
4.2.2	(m)	Attribute:	FromResponderToRequestor (Ordered)
4.3	(m)	Attribute:	Priority
4.3.1	(m)	Attribute:	DllPriority (Urgent, Normal, TimeAvailable)
4.3.2	(m)	Attribute:	DllPriorityNegotiated (Urgent, Normal, TimeAvailable)
4.4	(m)	Attribute:	DlpduAuthentication (Ordinary, Source, Maximal)
4.5	(m)	Attribute:	ResidualActivity
4.5.1	(m)	Attribute:	ResidualActivityAsSender (True, False)
4.5.2	(m)	Attribute:	ResidualActivityAsReceiver (True, False)
4.6	(m)	Attribute:	MaxConfirmDelay
4.6.1	(m)	Attribute:	MaxConfirmDelayOnDlConnect
4.6.2	(m)	Attribute:	MaxConfirmDelayOnDlData
4.7	(m)	Attribute:	DlSchedulingPolicy (Explicit)
4.8	(m)	Attribute:	MaxDlsduSizes
4.8.1	(m)	Attribute:	MaxDlsduSizeFromRequestor
4.8.2	(m)	Attribute:	MaxDlsduSizeFromResponder
4.8.3	(m)	Attribute:	MaxDlsduSizeFromRequestorNegotiated
4.8.4	(m)	Attribute:	MaxDlsduSizeFromResponderNegotiated

DLL SERVICES:

1	(m)	OpsService:	DL-Put
2	(m)	OpsService:	DL-Get
3	(m)	OpsService:	DL-Connect
4	(m)	OpsService:	DL-Connection-Established
5	(m)	OpsService:	DL-Disconnect
6	(m)	OpsService:	DL-Buffer-Received
7	(m)	OpsService:	DL-Buffer-Sent

M.2.1 Attributes

M.2.1.1 LocalDlcepAddress

This attribute specifies the local DLCEP address and identifies the DLCEP. The value of this attribute is used as the “DLCEP-address” parameter of the DLL.

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

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

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

M.2.1.2 RemoteDlcepAddress

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

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

NOTE 1 The value of this attribute is also carried in the header part of the Establish Request PDU.

This attribute contains the following three subattributes: Link Address, Node Address, and Selector.

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

M.2.1.3 DlsapRole

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

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

M.2.1.4 QosParameterSet

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

M.2.1.4.1 DlcepClass

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

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

M.2.1.4.2 DlcepDataDeliveryFeatures

These attributes specify the data delivery features of the DLL.

The permitted value Ordered corresponds to ORDERED defined by the Data Link Layer specification.

NOTE The FromRequestorToResponder and FromResponderToRequestor attributes shall have the same value.

M.2.1.4.2.1 FromRequestorToResponder

This attribute specifies the DLL data delivery feature of the AREP as a sender. It supplies the value for the “DLCEP data delivery features from requestor to responder(s)” parameter defined in the DLL.

M.2.1.4.2.2 FromResponderToRequestor

This attribute specifies the DLL data delivery feature of the AREP as a receiver. It supplies the value for the “DLCEP data delivery features from responder(s) to requestor” parameter defined in the DLL.

M.2.1.4.3 Priority

M.2.1.4.3.1 DllPriority

This attribute specifies the DLL priority before negotiation.

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

M.2.1.4.3.2 DllPriorityNegotiated

This attribute specifies the DLL priority after negotiation.

M.2.1.4.4 DlpduAuthentication

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

This attribute supplies the value for the “DLPDU-authentication” parameter of the DLL. The permitted value Ordinary, Source and Maximal correspond to ORDINARY, SOURCE and MAXIMAL respectively, as defined in the Fieldbus Data Link Layer specification.

M.2.1.4.5 ResidualActivityAsSender

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

M.2.1.4.5.1 ResidualActivityAsReceiver

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

M.2.1.4.6 MaxConfirmDelay

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

M.2.1.4.6.1 MaxConfirmDelayOnDlConnect

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

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

M.2.1.4.6.2 MaxConfirmDelayOnDlData

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

This attribute supplies the value for the “Maximum confirmation delay on DL-Data” parameter specified in the Data Link Layer specification (IEC 61158-3).

M.2.1.4.7 DlSchedulingPolicy

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

This attribute supplies the value for the “DL-Scheduling-policy” parameter of the DLL. The permitted value Explicit corresponds to EXPLICIT defined in the Fieldbus Data Link Layer specification.

M.2.1.4.8 MaxDlsduSize

M.2.1.4.8.1 MaxDlsduSizeFromRequestor

This attribute specifies the maximum length of an FAL-PDU that can be sent from this AREP before negotiation.

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

M.2.1.4.8.2 MaxDlsduSizeFromResponder

This attribute specifies the maximum length of an FAL-PDU that can be received by this AREP before negotiation.

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

M.2.1.4.8.3 MaxDlsduSizeFromRequestorNegotiated

This attribute specifies the maximum length of an FAL-PDU that can be sent from this AREP after negotiation.

M.2.1.4.8.4 MaxDlsduSizeFromResponderNegotiated

This attribute specifies the maximum length of an FAL-PDU that can be received by this AREP after negotiation.

M.2.2 DLL Services

Refer to annex C for DLL service description.

M.3 BNB ARPM State Machine

M.3.1 BNB ARPM States