

BS EN 61158-6-9:2014



BSI Standards Publication

Industrial communication networks — Fieldbus specifications

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

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 61158-6-9:2014. It is identical to IEC 61158-6-9:2014. It supersedes BS EN 61158-6-9:2012 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications: process measurement and control, including fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2014.
Published by BSI Standards Limited 2014

ISBN 978 0 580 79472 8
ICS 25.040.40; 35.100.70; 35.110

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 November 2014.

Amendments issued since publication

Date	Text affected
-------------	----------------------

EUROPEAN STANDARD

EN 61158-6-9

NORME EUROPÉENNE

EUROPÄISCHE NORM

October 2014

ICS 25.040.40; 35.100.70; 35.110

Supersedes EN 61158-6-9:2012

English Version

Industrial communication networks - Fieldbus specifications -
Part 6-9: Application layer protocol specification - Type 9
elements
(IEC 61158-6-9:2014)

Réseaux de communication industriels - Spécifications des
bus de terrain - Partie 6-9: Spécification du protocole de la
couche application - Éléments de type 9
(CEI 61158-6-9:2014)

Industrielle Kommunikationsnetze - Feldbusse - Teil 6-9:
Protokollspezifikation des Application Layer
(Anwendungsschicht) - Typ 9-Elemente
(IEC 61158-6-9:2014)

This European Standard was approved by CENELEC on 2014-09-23. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.



European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Foreword

The text of document 65C/764/FDIS, future edition 3 of IEC 61158-6-9, prepared by SC 65C "Industrial networks" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61158-6-9:2014.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2015-06-23
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2017-09-23

This document supersedes EN 61158-6-9:2012.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

Endorsement notice

The text of the International Standard IEC 61158-6-9:2014 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61784-1	NOTE	Harmonized as EN 61784-1.
IEC 61784-2	NOTE	Harmonized as EN 61784-2.

Annex ZA (normative)

Normative references to international publications with their corresponding European publications

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

NOTE 1 When an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 61158-1	-	Industrial communication networks - Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series	EN 61158-1	-
IEC 61158-3-1	-	Industrial communication networks - Fieldbus specifications - Part 3-1: Data-link layer service definition - Type 1 elements	EN 61158-3-1	-
IEC 61158-4-1	-	Industrial communication networks - Fieldbus specifications - Part 4-1: Data-link layer protocol specification - Type 1 elements	EN 61158-4-1	-
IEC 61158-5-5	-	Industrial communication networks - Fieldbus specifications - Part 5-5: Application layer service definition - Type 5 elements	EN 61158-5-5	-
IEC 61158-5-9	-	Industrial communication networks - Fieldbus specifications - Part 5-9: Application layer service definition - Type 9 elements	EN 61158-5-9	-
ISO/IEC 646	-	Information technology - ISO 7-bit coded character set for information interchange	-	-
ISO/IEC 7498-1	-	Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model	-	-
ISO/IEC 8824-1	-	Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation	-	-

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
ISO/IEC 8825-1	-	Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)	-	-
ISO/IEC 9545	-	Information technology - Open Systems Interconnection - Application Layer structure	-	-
ISO/IEC 10731	-	Information technology - Open Systems Interconnection - Basic Reference Model - Conventions for the definition of OSI services	-	-
ISO/IEC/IEEE 60559	-	Information technology - Microprocessor Systems - Floating-Point arithmetic	-	-

CONTENTS

INTRODUCTION.....	7
1 Scope.....	8
1.1 General.....	8
1.2 Specifications.....	8
1.3 Conformance.....	9
2 Normative references	9
3 Terms, definitions, symbols, abbreviations and conventions	10
3.1 Terms and definitions from other ISO/IEC standards	10
3.2 IEC 61158-1 terms	11
3.3 Abbreviations and symbols.....	14
3.4 Conventions	15
3.5 Conventions used in state machines	16
4 Abstract syntax.....	17
4.1 FAL-AR PDU abstract syntax	17
4.2 Abstract syntax of PDUBody.....	19
4.3 Type definitions for ASEs	22
4.4 Abstract syntax of data types	27
5 Transfer syntax	28
6 Structure of FAL protocol state machines	39
7 AP-Context state machines	40
7.1 VCR PM structure	40
7.2 VCR PM state machine	41
8 FAL service protocol machine (FSPM).....	53
8.1 General.....	53
8.2 FSPM state tables.....	53
8.3 Functions used by FSPM.....	56
8.4 Parameters of FSPM/ARPM primitives	56
9 Application relationship protocol machines (ARPMs)	56
9.1 AREP mapping to data-link layer	56
9.2 Application relationship protocol machines (ARPMs)	66
9.3 AREP state machine primitive definitions	82
9.4 AREP state machine functions	83
10 DLL mapping protocol machine (DMPM).....	84
10.1 DMPM States	84
10.2 DMPM state table.....	85
10.3 Primitives exchanged between data-link layer and DMPM	91
10.4 Functions used by DMPM.....	93
Bibliography.....	95
Figure 1 – Insertion of identification information in the FMS PDU.....	29
Figure 2 – Identification	30
Figure 3 – Coding with identification	31
Figure 4 – Coding without identification	31
Figure 5 – Representation of the value true	31

Figure 6 – Representation of the value false	31
Figure 7 – Coding of data of data type Integer16	32
Figure 8 – Coding of data of data type Unsigned16	32
Figure 9 – Coding of data of data type Floating Point	33
Figure 10 – Coding of data of data type Visible String	33
Figure 11 – Coding of data of data type Octet String	34
Figure 12 – Coding of data of type Date	34
Figure 13 – Coding of data of data type Time-of-day	35
Figure 14 – Coding of data of data type Time-difference	36
Figure 15 – Coding of data of data type Bit String	36
Figure 16 – Coding of data of data type Time-value	37
Figure 17 – Coding of data of user data definitions with identifier	37
Figure 18 – Coding of data of user data definitions without identifier	37
Figure 19 – Coding of ID info for a SEQUENCE	38
Figure 20 – Relationships among protocol machines and adjacent layers	39
Figure 21 – Relationships among protocol machines and adjacent layers	40
Figure 22 – VCR state machine	41
Figure 23 – State transition diagram of FSPM	53
Figure 24 – State transition diagram of the QUU ARPM	67
Figure 25 – State transition diagram of QUB ARPM	69
Figure 26 – State transition diagram of the BNU ARPM	77
Figure 27 – State transition diagram of DMPM	85
Table 1 – Conventions used for state machines	16
Table 2 – Coding for Date type	34
Table 3 – AP-VCR state machine transactions	42
Table 4 – Primitives issued by FAL-User to VCR PM	51
Table 5 – Primitives issued by VCR PM to FAL-User	51
Table 6 – Primitives issued by VCR PM to FSPM	52
Table 7 – Primitives issued by FSPM to VCR PM	52
Table 8 – FSPM state table – sender transactions	54
Table 9 – FSPM state table – receiver transactions	55
Table 10 – Function SelectArep()	56
Table 11 – Parameters used with primitives exchanged between FSPM and ARPM	56
Table 12 – QUU ARPM states	67
Table 13 – QUU ARPM state table – sender transactions	67
Table 14 – QUU ARPM state table – receiver transactions	68
Table 15 – QUB ARPM states	68
Table 16 – QUB ARPM state table – sender transactions	69
Table 17 – QUB ARPM state table – receiver transactions	71
Table 18 – BNU ARPM states	77
Table 19 – BNU ARPM state table – sender transactions	78
Table 20 – BNU ARPM state table – receiver transactions	79

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

INTRODUCTION

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

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

1 Scope

1.1 General

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

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

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

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

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

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

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

1.2 Specifications

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

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of protocols standardized in IEC 61158-6.

1.3 Conformance

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

2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 Terms and definitions from ISO/IEC 7498-1

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

3.1.2 Terms and definitions from ISO/IEC 9545

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

3.1.3 Terms and definitions from ISO/IEC 8824-1

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

3.1.4 Terms and definitions from ISO/IEC 8825-1

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

3.2 IEC 61158-1 terms

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

3.2.1

application

function or data structure for which data is consumed or produced

3.2.2

application layer interoperability

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

3.2.3

application object

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

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

3.2.4

application process

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

3.2.5

application process identifier

distinguishes multiple application processes used in a device

3.2.6

application process object

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

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

3.2.7

application process object class

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

3.2.8

application relationship

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

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

3.2.9

application relationship application service element

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

3.2.10**application relationship endpoint**

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

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

3.2.11**attribute**

description of an externally visible characteristic or feature of an object

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

3.2.12**behaviour**

indication of how the object responds to particular events

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

3.2.13**class**

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

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

3.2.14**class attributes**

attribute that is shared by all objects within the same class

3.2.15**class code**

unique identifier assigned to each object class

3.2.16**class specific service**

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

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

3.2.17**client**

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

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

3.2.18**conveyance path**

unidirectional flow of APDUs across an application relationship

3.2.19**cyclic**

events which repeat in a regular and repetitive manner

3.2.20**dedicated AR**

AR used directly by the FAL User

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

3.2.21**device**

physical hardware connection to the link

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

3.2.22**device profile**

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

3.2.23**dynamic AR**

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

3.2.24**endpoint**

one of the communicating entities involved in a connection

3.2.25**error**

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

3.2.26**error class**

general grouping for error definitions

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

3.2.27**error code**

identification of a specific type of error within an error class

3.2.28**FAL subnet**

networks composed of one or more data link segments

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

3.2.29**logical device**

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

3.2.30**management information**

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

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

3.2.31**network**

series of nodes connected by some type of communication medium

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

3.2.32**peer**

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

3.2.33**pre-defined AR endpoint**

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

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

3.2.34**pre-established AR endpoint**

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

3.2.35**publisher**

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

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

3.2.36**server**

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

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

3.2.37**service**

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

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

3.2.38**subscriber**

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

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

3.3 Abbreviations and symbols

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

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

3.4 Conventions

3.4.1 General concept

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

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

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

3.4.2 Conventions for class definitions

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

3.4.3 Abstract syntax conventions

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

3.5 Conventions used in state machines

The state machines are described in Table 1:

Table 1 – Conventions used for state machines

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

The conventions used in the state machines are as follows:

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

`xxx` A parameter name.

Example:

`Identifier := reason`

means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'

`"xxx"` Indicates fixed value.

Example:

`Identifier := "abc"`

means value "abc" is assigned to a parameter named 'Identifier.'

`=` A logical condition to indicate an item on the left is equal to an item on the right.

`<` A logical condition to indicate an item on the left is less than the item on the right.

`>` A logical condition to indicate an item on the left is greater than the item on the right.

`<>` A logical condition to indicate an item on the left is not equal to an item on the right.

`&&` Logical "AND"

`||` Logical "OR"

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

Example:

`for (Identifier := start_value to end_value)`

`actions`

`endfor`

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

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

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

4 Abstract syntax

4.1 FAL-AR PDU abstract syntax

4.1.1 Top level definition

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

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

4.1.2 Confirmed send service

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

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

4.1.3 Unconfirmed send service

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

4.1.4 Unconfirmed acknowledge send service

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

4.1.5 InvokeID

InvokeID ::= Unsigned8

4.1.6 Establish service

MaxOSCC ::= Unsigned8

MaxOSCS ::= Unsigned8

MaxUCSC ::= Unsigned8

MaxUCSS ::= Unsigned8

CIU ::= Unsigned32

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

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

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

4.1.7 ConType

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

4.1.8 Data send acknowledge service

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

4.1.9 Protocol-code

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

4.1.10 Block-number

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

4.1.11 Block-length

Block-Length ::= Unsigned

4.1.12 Protocol-data

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

4.1.13 Address2 ARP

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

4.1.14 Destination-address

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

4.1.15 Source-address

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

4.1.16 Destination-node

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

4.1.17 Source-node

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

4.1.18 Destination-subnode

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

4.1.19 Source-subnode

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

4.2 Abstract syntax of PDUBody**4.2.1 Abort service**

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

4.2.2 ConfirmedServiceRequest

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

4.2.3 ConfirmedServiceResponse

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

4.2.4 ConfirmedServiceError

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

4.2.5 Error type

Error type as specified in 4.1.4.6.

4.2.6 Error Fi type

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

4.2.7 Error class

Error Class as specified in 4.1.4.7.

4.2.8 Unconfirmed PDUs

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

```
UnconfirmedSendPD-PDU ::= BIT STRING
```

4.2.9 Management ASE

4.2.9.1 Get attributes service

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

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

4.2.10 Application process ASE

4.2.10.1 Get status service

Status-RequestPDU ::= NULL

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

4.2.10.2 Identify service

Identify-RequestPDU ::= NULL

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

4.2.10.3 Initiate service

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

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

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

4.2.10.4 Reject service

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


4.2.11 Function invocation ASE

4.2.11.1 Reset service

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

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

4.2.11.2 Resume service

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

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

4.2.11.3 Start service

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

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

4.2.11.4 Stop service

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

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

4.2.12 Variable ASE

4.2.12.1 Information report service

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

4.2.12.2 Read service

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

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

4.2.12.3 Write service

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

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

4.3 Type definitions for ASEs

4.3.1 AP ASE types

4.3.1.1 Ap_AccessProtectionSupported

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

4.3.1.2 Ap_AccessControl

```

Ap_AccessControl ::= BitString {
    password_Bit1          (8),
    password_Bit2          (7),
    password_Bit3          (6),
    password_Bit4          (5),
    password_Bit5          (4),
    password_Bit6          (3),
    password_Bit7          (2),
    password_Bit8          (1),
    access_Groups-1        (16),
    access_Groups-2        (15),
    access_Groups-3        (14),
    access_Groups-4        (13),
    access_Groups-5        (12),
    access_Groups-6        (11),
    access_Groups-7        (10),
    access_Groups-8        (9)
}

```

-- The Password (Unsigned8) is encoded as a bit string.

4.3.2 AR ASE types

4.3.2.1 Reason code and additional detail

4.3.2.1.1 Reason code

ReasonCode ::= Unsigned8

4.3.2.1.2 Additional detail

AdditionalDetail ::= OctetString

4.3.2.2 AREP

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

4.3.2.3 Abort type identifier

```

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

```

4.3.3 Function Invocation ASE types

4.3.3.1 Fi_AccessPrivilege

```

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

```

-- The Password (Unsigned8) is encoded as a bit string.

}

4.3.3.2 Fi_State

```

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

```

4.3.4 Management ASE types**4.3.4.1 Mn_PduSupportedMap**

```

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

```

4.3.5 Variable ASE types**4.3.5.1 Vr_AccessPrivilege**

```

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

```

4.3.6 General types**4.3.6.1 Gn_Deletable**

```

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

```

4.3.6.2 Gn_KeyAttribute

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

4.3.6.3 Gn_Length

```
Gn_Length ::= Unsigned8
```

4.3.6.4 Gn_MoreFollows

```
Gn_MoreFollows ::= Boolean
```

4.3.6.5 Gn_NumericID

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

4.3.6.6 Gn_Name

```
Gn_Name ::= OctetString
```

4.3.6.7 Gn_ObjectClass

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

4.3.6.8 Gn_ObjectDefinition

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

4.3.6.9 Gn_Reusable

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

4.3.6.10 Gn_SubIndex

```
Gn_SubIndex ::= Unsigned8
```

4.3.6.11 Gn_TypeDescription

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

4.3.7 Object definitions

4.3.7.1 Top Level Definition

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

4.3.7.2 ListHeader

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

4.3.7.3 DataTypeList

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

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

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

4.3.7.4 StaticList

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

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

```

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

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

```

4.3.7.5 FunctionInvocationDefinition

```

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

```

4.4 Abstract syntax of data types

4.4.1 Referenced data types

4.4.2 Notation for the Boolean type

```

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

```

4.4.3 Notation for the Integer types

```

Integer ::= INTEGER -- any integer
Integer8 ::= INTEGER (-128..+127) -- range  $-2^7 \leq i \leq 2^7-1$ 
Integer16 ::= INTEGER (-32768..+32767) -- range  $-2^{15} \leq i \leq 2^{15}-1$ 
Integer32 ::= INTEGER -- range  $-2^{31} \leq i \leq 2^{31}-1$ 

```

4.4.4 Notation for the Unsigned types

```

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

```

4.4.5 Notation for the Floating Point type

```

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

```

4.4.6 Notation for the BitString type

```

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

```

4.4.7 Notation for the OctetString type

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

4.4.8 Notation for VisibleString type

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

4.4.9 Notation for BinaryDate type

BinaryDate ::= OctetString7

4.4.10 Notation for TimeOfDay type

TimeOfDay ::= OctetString6

4.4.11 Notation for TimeDifference type

TimeDifference ::= OctetString6

4.4.12 Notation for TimeValue type

TimeValue ::= OctetString8

4.4.13 Notation for DL—Time-offset type

DL-Time-offsetType ::= OctetString

5 Transfer syntax

5.1.1 General

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

The conditions in the fieldbus area are the following:

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

5.1.2 Coding rules

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

Structure of a PDU is shown in Figure 1.

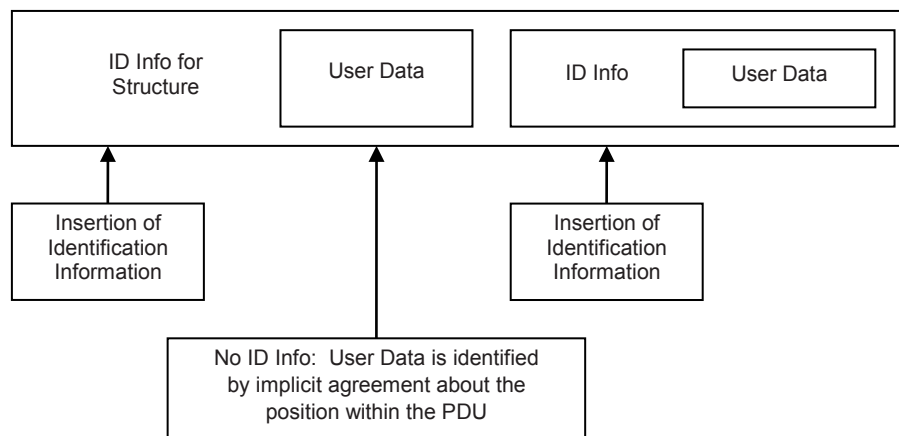


Figure 1 – Insertion of identification information in the FMS PDU

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

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

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

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

5.1.3 Structure of the identification information

5.1.3.1 General

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

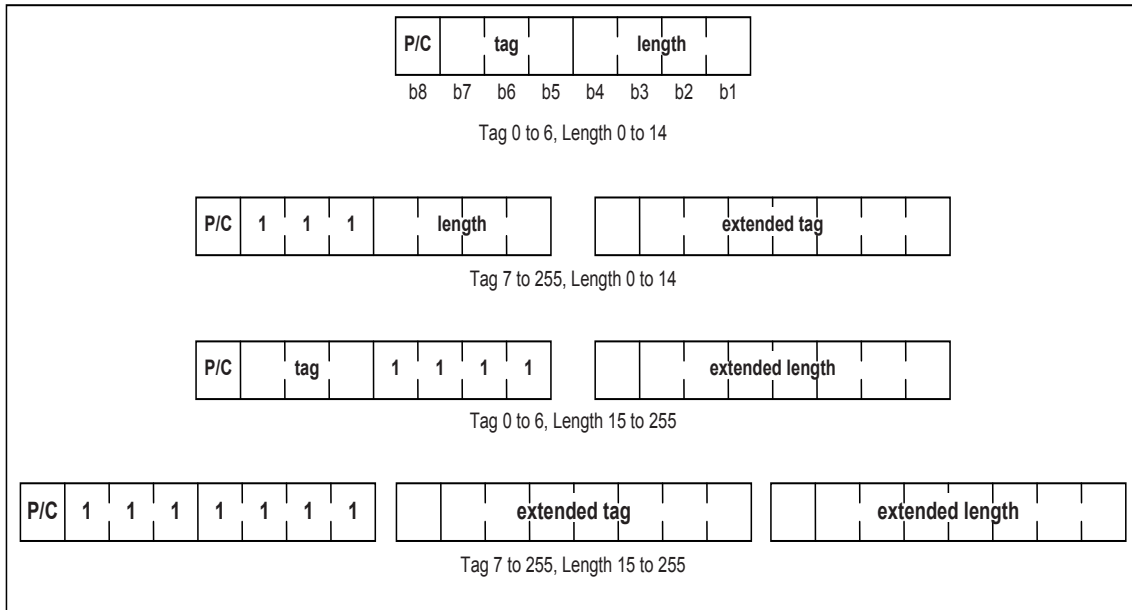


Figure 2 – Identification

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

P/C Flag =0 <=> simple component

P/C Flag =1 <=> structured component

The tag identifies the semantics of the component.

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

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

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

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

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

5.1.3.2 Data types

5.1.3.2.1 General

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

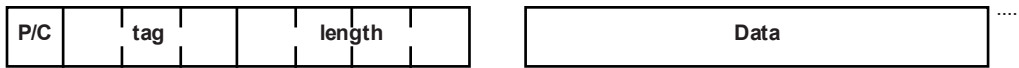


Figure 3 – Coding with identification

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

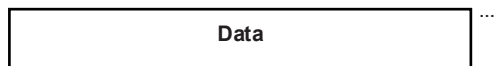


Figure 4 – Coding without identification

5.1.3.2.2 Coding for Boolean type

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

Notation: Range of values:		Boolean										
Coding:		true or false false is represented by the value 0, true is represented by the value FF										
bits	8	7	6	5	4	3	2	1				
octet	1 1 1 1 1 1 1 1											

Figure 5 – Representation of the value true

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

Figure 6 – Representation of the value false

5.1.3.2.3 Coding for Integer types

Integer values are signed quantities as shown in Figure 7.

Notation:		Integer8, Integer16, Integer32								
Range of Values:		Data type	range of values						length	
		Integer8	$-128 \leq i \leq 127$						1 octet	
		Integer16	$-32768 \leq i \leq 32767$						2 octets	
		Integer32	$-2^{31} \leq i \leq 2^{31} - 1$						4 octets	
Coding:		In two's complement representation the MSB (Most Significant Bit) is the bit after the sign bit (SN) in the first octet. SN = 0: positive numbers and z SN = 1: negative numbers								
bits		8	7	6	5	4	3	2	1	
octets										
1		SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
2		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Figure 7 – Coding of data of data type Integer16

5.1.3.2.4 Coding for Unsigned types

Unsigned Values are encoded as shown in Figure 8.

Notation:		Unsigned8, Unsigned16, Unsigned32								
Range of Values:		Data type	range of values						length	
		Unsigned8	$0 \leq i \leq 255$						1 octet	
		Unsigned16	$0 \leq i \leq 65\,535$						2 octets	
		Unsigned32	$0 \leq i \leq 4\,294\,967\,295$						4 octets	
Coding		Binary								
bits		8	7	6	5	4	3	2	1	
octets										
1		2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
2		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Figure 8 – Coding of data of data type Unsigned16

5.1.3.2.5 Coding for Floating Point type

Floating Point values are encoded as shown in Figure 9.

Notation:		Floating-Point (4 octet)							
Range of Values:		see IEEE Std 754 Short Real Number (32 bits)							
Coding:		see IEEE Std 754 Short Real Number (32 bits)							
		LSB							
bits	8	7	6	5	4	3	2	1	
octets	Exponent (E)								
1	SN	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	
2	(E) 2 ⁰	Fraction (F)							
3	Fraction (F)								
	2 ⁻⁸	2 ⁻⁹	2 ⁻¹⁰	2 ⁻¹¹	2 ⁻¹²	2 ⁻¹³	2 ⁻¹⁴	2 ⁻¹⁵	
4	Fraction (F)								
	2 ⁻¹⁶	2 ⁻¹⁷	2 ⁻¹⁸	2 ⁻¹⁹	2 ⁻²⁰	2 ⁻²¹	2 ⁻²²	2 ⁻²³	
SN: sign 0 = positive, 1 = negative									

Figure 9 – Coding of data of data type Floating Point

5.1.3.2.6 Coding for Visible String type

Visible String values are encoded as shown in Figure 10.

Notation:		Visible-String							
Range of Values:		see ISO/IEC 646 and ISO/IEC 2375: Defining							
registration									
		number 2 + SPACE							
Coding:		see ISO/IEC 646							
bits	8	7	6	5	4	3	2	1	
octets									
1	first character								
2	second character								
.	etc.								
.									
n									

Figure 10 – Coding of data of data type Visible String

5.1.3.2.7 Coding for Octet String type

Octet String values are encoded as shown in Figure 11.

Notation: Octet String
Coding: Binary

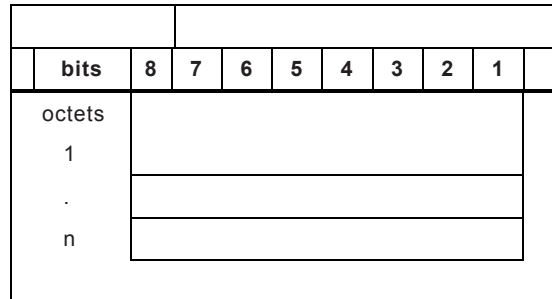


Figure 11 – Coding of data of data type Octet String

5.1.3.2.8 Coding for Date type

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

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

Table 2 – Coding for Date type

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

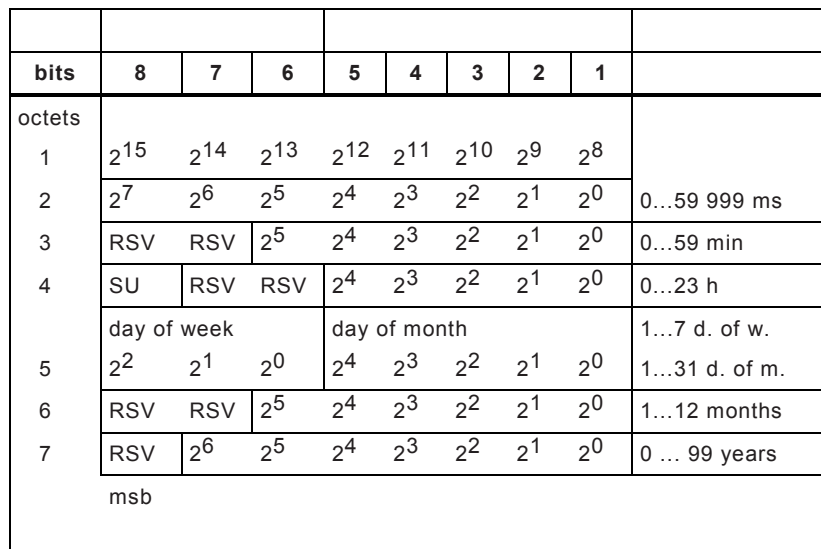


Figure 12 – Coding of data of type Date

5.1.3.2.9 Coding for Time-of-day type

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

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

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

Notation: Time-of-day

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

Coding:

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

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

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

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

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

5.1.3.2.10 Coding for Time-difference type

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

Notation Time-difference

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

Coding:

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

bits	8	7	6	5	4	3	2	1	
octets									number of milliseconds
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	
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	number of days optional
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	optional
msb									

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

5.1.3.2.11 Coding for Bit String type

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

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

Notation: Bit String
Coding: Binary

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

Figure 15 – Coding of data of data type Bit String

5.1.3.2.12 Coding for Time-value type

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

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

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

5.1.3.3 User data definitions

5.1.3.3.1 General

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

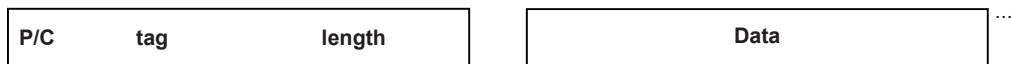


Figure 17 – Coding of data of user data definitions with identifier

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

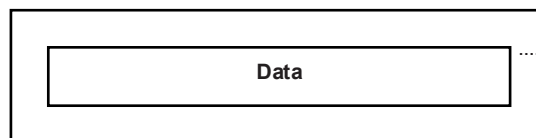


Figure 18 – Coding of data of user data definitions without identifier

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

5.1.3.3.2 Coding for Null

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

5.1.3.3.3 Coding for Packed

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

5.1.3.4 Coding of structure information

5.1.3.4.1 General

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

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

5.1.3.4.2 SEQUENCE

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

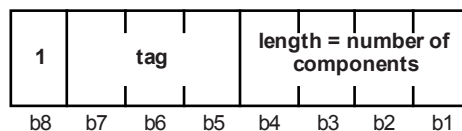


Figure 19 – Coding of ID info for a SEQUENCE

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

Example:

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

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

5.1.3.4.3 SEQUENCE OF

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

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

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

Example:

```

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

5.1.3.4.4 CHOICE

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

Example:

```

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

6 Structure of FAL protocol state machines

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

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

The behavior of the FAL is described by three integrated protocol machines. Specific sets of these protocol machines are defined for different AREP types. The three protocol machines are: FAL service Protocol Machine (FSPM), the Application Relationship Protocol Machine (ARPM), and the data-link layer Mapping Protocol Machine (DMPM). The relationship among these protocol machines as well as primitives exchanged among them are depicted in Figure 20.

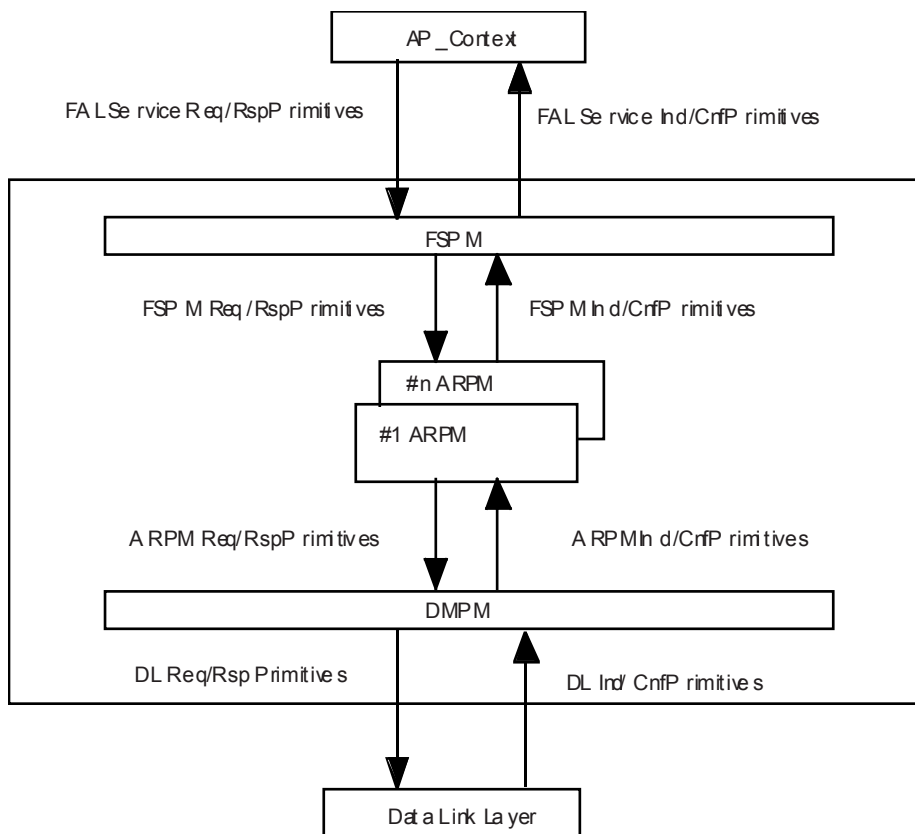


Figure 20 – Relationships among protocol machines and adjacent layers

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

- a) to accept service primitives from the FAL service user and convert them into FAL internal primitives;
- b) to select an appropriate ARPM state machine based on the AREP Identifier parameter supplied by the AP-Context and send FAL internal primitives to the selected ARPM;
- c) to accept FAL internal primitives from the ARPM and convert them into service primitives for the AP-Context;
- d) to deliver the FAL service primitives to the AP-Context based on the AREP Identifier parameter associated with the primitives.

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

- e) to accept FAL internal primitives from the FSPM and create and send other FAL internal primitives to either the FSPM or the DMPM, based on the AREP and primitive types;
- f) to accept FAL internal primitives from the DMPM and send them to the FSPM as a form of FAL internal primitives;
- g) if the primitives are for the Establish or Abort service, it shall try to establish or release the specified AR.

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

- h) to accept FAL internal primitives from the ARPM, prepare DLL service primitives, and send them to the DLL;
- i) to receive DLL indication or confirmation primitives from the DLL and send them to the ARPM in a form of FAL internal primitives.

7 AP-Context state machines

7.1 VCR PM structure

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

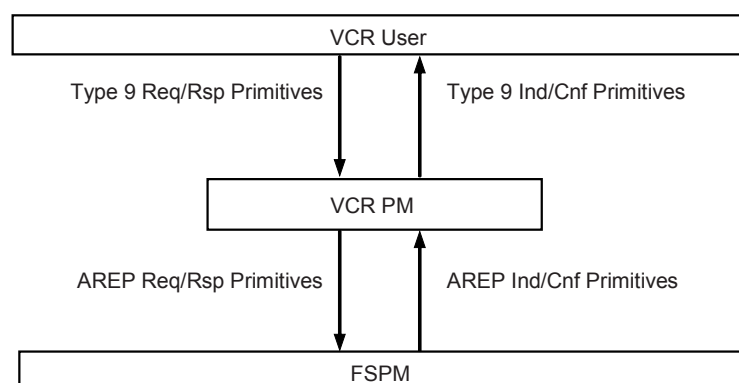


Figure 21 – Relationships among protocol machines and adjacent layers

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

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

7.2 VCR PM state machine

7.2.1 General

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

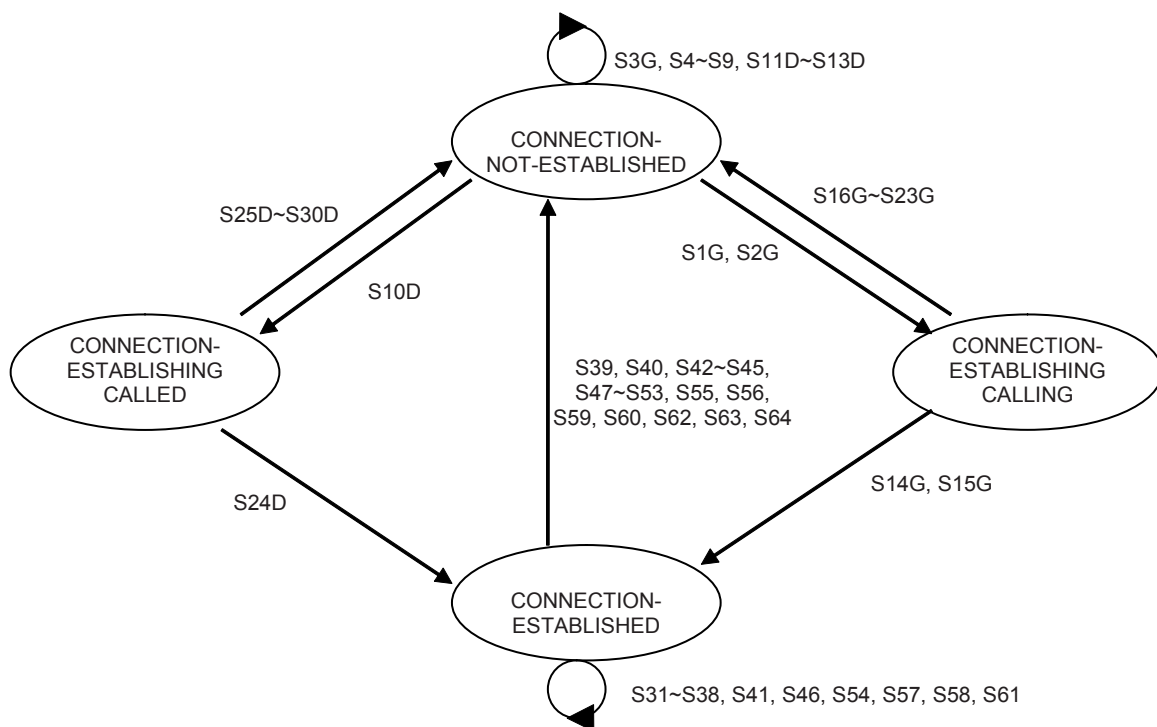


Figure 22 – VCR state machine

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

7.2.2 AP-VCR states

CONNECTION-NOT-ESTABLISHED

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

CONNECTION-ESTABLISHING (CALLING)

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

CONNECTION-ESTABLISHING (CALLED)

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

CONNECTION-ESTABLISHED

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

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

7.2.3 AP-VCR initiation state transitions

Table 3 defines the AP-VCR state transitions.

Table 3 – AP-VCR state machine transactions

#	Current state	Event or condition => action	Next state
S1G	CONNECTION-NOT-ESTABLISHED	Initiate.req && VCR is valid && VCR type = QUB => ASC.req{ Data := Initiate-RequestPDU }	CONNECTION – ESTABLISHING-CALLING
S2G	CONNECTION-NOT-ESTABLISHED	Initiate.req && VCR is valid && VCR type <> QUB => ASC.req{ Data := NULL }	CONNECTION – ESTABLISHING-CALLING
S3G	CONNECTION-NOT-ESTABLISHED	Initiate.req && VCR is not valid => Abort.ind{ AbortIdentifier := FMS, ReasonCode := VCR error }	CONNECTION-NOT-ESTABLISHED
S4	CONNECTION-NOT-ESTABLISHED	Abort.req => (no actions taken)	CONNECTION-NOT-ESTABLISHED
S5	CONNECTION-NOT-ESTABLISHED	FMS service.primitive other than Initiate & Abort => Abort.ind{ AbortIdentifier := FMS, ReasonCode := User error }	CONNECTION-NOT-ESTABLISHED
S6	CONNECTION-NOT-ESTABLISHED	ABT.ind => (no actions taken)	CONNECTION-NOT-ESTABLISHED
S7	CONNECTION-NOT-ESTABLISHED	Faulty or unknown FAS service primitive => ABT.req{ AbortIdentifier := FMS, ReasonCode := FAS error }	CONNECTION-NOT-ESTABLISHED
S8	CONNECTION-NOT-ESTABLISHED	FAS service primitive other than ASC.ind and ABT.ind => ABT.req{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S9	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = not allowed, unknown or faulty FMS PDU => ABT.req{ AbortIdentifier := FMS, ReasonCode := FMS PDU error }	CONNECTION-NOT-ESTABLISHED
S10D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data Initiate-RequestPDU && VCR is valid && Max FMS PDU length test is positive && FmsFeaturesSupported test is positive => Initiate.ind{ }	CONNECTION – ESTABLISHING-CALLED
S11D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = Initiate-RequestPDU && VCR is invalid => ABT.req{ AbortIdentifier := FMS, ReasonCode := VCR error }	CONNECTION-NOT-ESTABLISHED
S12D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = Initiate-RequestPDU && VCR is valid && Max FMS PDU length test is negative => ASC.rsp(-){ Data := Initiate-ErrorPDU{ ErrorCode := max-fms-pdu-size-insufficient } }	CONNECTION-NOT-ESTABLISHED
S13D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = Initiate-RequestPDU && VCR is valid && Max FMS PDU length test is positive && FmsFeaturesSupported test is negative => ASC.rsp(-){ Data := Initiate-ErrorPDU{ ErrorCode := feature-not-supported } }	CONNECTION-NOT-ESTABLISHED
S14G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(+) && Data = Initiate-ResponsePDU && VCR type = QUB => Initiate.cnf(+){ }	CONNECTION-ESTABLISHED
S15G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(+) && Data = NULL && VCR type = QUU or BNU => Initiate.cnf(+){ }	CONNECTION-ESTABLISHED
S16G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(-) && Data = Initiate-ErrorPDU && VCR type = QUB => Initiate.cnf(-){ ErrorCode := Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED
S17G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf(-) && Data = NULL && VCR type = QUU or BNU => Initiate.cnf(-){ ErrorCode := Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S18G	CONNECTION - ESTABLISHING -CALLING	ABT.ind => Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED
S19G	CONNECTION - ESTABLISHING -CALLING	Abort.req => ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED
S20G	CONNECTION - ESTABLISHING -CALLING	Faulty or unknown FAS service primitive => ABT.req{ AbortIdentifier := FMS, ReasonCode := FAS error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FAS error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S21G	CONNECTION - ESTABLISHING -CALLING	FAS service primitive other than ASC.cnf and ABT.ind => ABT.req{ AbortIdentifier := FMS, ReasonCode :=Connection state conflict FAS }, Abort.ind{ AbortIdentifier := FMS, ReasonCode :=Connection state conflict FAS }, reset VCR	CONNECTION-NOT-ESTABLISHED
S22G	CONNECTION - ESTABLISHING -CALLING	ASC.cnf && Data = not allowed, unknown or faulty FMS PDU => ABT.req{ AbortIdentifier := FMS, ReasonCode := FMS PDU error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FMS PDU error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S23G	CONNECTION - ESTABLISHING -CALLING	FMS service.primitive other than Initiate.rsp and Abort.req => ABT.req{ AbortIdentifier := FMS, ReasonCode := User error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := User error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S24D	CONNECTION - ESTABLISHING -CALLED	Initiate.rsp(+) => ASC.rsp(+){ Data = Initiate-ResponsePDU }	CONNECTION-ESTABLISHED
S25D	CONNECTION - ESTABLISHING -CALLED	Initiate.rsp(-) => ASC.rsp(-){ Data = Initiate-ErrorPDU{ ErrorCode = ErrorCode of Initiate.rsp } }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S26D	CONNECTION - ESTABLISHING - CALLED	Abort.req => ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION - NOT - ESTABLISHED
S27D	CONNECTION - ESTABLISHING - CALLED	ABT.ind => Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION - NOT - ESTABLISHED
S28D	CONNECTION - ESTABLISHING - CALLED	Faulty or unknown FAS service primitive => ABT.req{ AbortIdentifier := FMS, ReasonCode := FAS error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FAS error }, reset VCR	CONNECTION - NOT - ESTABLISHED
S29D	CONNECTION - ESTABLISHING - CALLED	FAS service primitive other ABT.ind => ABT.req{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }, reset VCR	CONNECTION - NOT - ESTABLISHED
S30D	CONNECTION - ESTABLISHING - CALLED	FMS service.primitive other than Initiate.rsp and Abort.req => ABT.req{ AbortIdentifier := FMS, ReasonCode := User error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := User error }, reset VCR	CONNECTION - NOT - ESTABLISHED
S31	CONNECTION - ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) positive => DTC.req{ Data := Confirmed-RequestPDU }, OSCS := OSCS + 1	CONNECTION - ESTABLISHED
S32	CONNECTION - ESTABLISHED	ConfirmedService.req && OSCS = ActualMaxSCC => Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := Max-services-overflow }	CONNECTION - ESTABLISHED

#	Current state	Event or condition => action	Next state
S33	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID already existent => Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := Invoke-ID-exists }	CONNECTION-ESTABLISHED
S34	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length > Max FMS PDU sending => Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := PDU-size }	CONNECTION-ESTABLISHED
S35	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) negative => Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := Feature-not-supported-connection-oriented }	CONNECTION-ESTABLISHED
S36	CONNECTION-ESTABLISHED	UnconfirmedService.req && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) positive => DTU.req{ Data := Unconfirmed-PDU }	CONNECTION-ESTABLISHED
S37	CONNECTION-ESTABLISHED	UnconfirmedService.req && PDU length > Max FMS PDU sending => Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in UnconfirmedService.req, RejectPDUType := Unconfirmed-PDU, RejectCode := PDU-size }	CONNECTION-ESTABLISHED
S38	CONNECTION-ESTABLISHED	UnconfirmedService.req && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) negative => Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in UnconfirmedService.req, RejectPDUType := Unconfirmed-PDU, RejectCode := Feature-not-supported-connection-oriented }	CONNECTION-ESTABLISHED
S39	CONNECTION-ESTABLISHED	Abort.req => ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S40	CONNECTION-ESTABLISHED	Faulty, unknown or not-allowed FMS service.primitive => ABT.req{ AbortIdentifier := FMS, ReasonCode := User error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := User error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S41	CONNECTION-ESTABLISHED	DTC.ind && Data = Confirmed-ServicePDU && PDU length ≤ Max FMS PDU receiving && OSCR < ActualMaxRCC && InvokeID not existent && Features Supported test (server) positive => ConfirmedService.ind{ } OSCR := OSCR + 1	CONNECTION-ESTABLISHED
S42	CONNECTION-ESTABLISHED	DTC.ind && Data = Confirmed-ServicePDU && PDU length > Max FMS PDU receiving => ABT.req{ AbortIdentifier := FMS, ReasonCode := PDU-size }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := PDU-size }, reset VCR	CONNECTION-NOT-ESTABLISHED
S43	CONNECTION-ESTABLISHED	DTC.ind && Data = Confirmed-ServicePDU && PDU length ≤ Max FMS PDU receiving && OSCR ≥ ActualMaxRCC => ABT.req{ AbortIdentifier := FMS, ReasonCode := Max-services-overflow }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Max-services-overflow }, reset VCR	CONNECTION-NOT-ESTABLISHED
S44	CONNECTION-ESTABLISHED	DTC.ind && Data = Confirmed-ServicePDU && PDU length ≤ Max FMS PDU receiving && OSCR < ActualMaxRCC && InvokeID already existent => ABT.req{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-request }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-request }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S45	CONNECTION-ESTABLISHED	DTC.ind && Data = Confirmed-ServicePDU && PDU length ≤ Max FMS PDU receiving && OSCRC < ActualMaxRCC && InvokeID not existent && Features Supported test (server) negative => ABT.req{ AbortIdentifier := FMS, ReasonCode := Feature-not-supported }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Feature-not-supported }, reset VCR	CONNECTION-NOT-ESTABLISHED
S46	CONNECTION-ESTABLISHED	DTU.ind && Data = Unconfirmed-PDU && PDU length ≤ Max FMS PDU receiving && Features Supported test (server) positive => UnconfirmedService.ind{ }	CONNECTION-ESTABLISHED
S47	CONNECTION-ESTABLISHED	DTU.ind && Data = Unconfirmed-PDU && PDU length > Max FMS PDU receiving => ABT.req{ AbortIdentifier := FMS, ReasonCode := PDU-size }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := PDU-size }, reset VCR	CONNECTION-NOT-ESTABLISHED
S48	CONNECTION-ESTABLISHED	DTU.ind && Data = Unconfirmed-PDU && PDU length ≤ Max FMS PDU receiving && Features Supported test (server) negative => ABT.req{ AbortIdentifier := FMS, ReasonCode := Feature-not-supported }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Feature-not-supported }, reset VCR	CONNECTION-NOT-ESTABLISHED
S49	CONNECTION-ESTABLISHED	ABT.ind => Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED
S50	CONNECTION-ESTABLISHED	ASC.ind && Data = Initiate-RequestPDU => ABT.req{ AbortIdentifier := FMS, ReasonCode := Connection-state-conflict-FMS }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Connection-state-conflict-FMS }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S51	CONNECTION-ESTABLISHED	Faulty or unknown FAS service primitive => <pre> ABT.req{ AbortIdentifier := FMS, ReasonCode := FAS error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FAS error }, reset VCR </pre>	CONNECTION-NOT-ESTABLISHED
S52	CONNECTION-ESTABLISHED	Not-allowed FAS service primitive => <pre> ABT.req{ AbortIdentifier := FMS, ReasonCode := Connection-state-conflict-FAS }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Connection-state-conflict-FAS }, reset VCR </pre>	CONNECTION-NOT-ESTABLISHED
S53	CONNECTION-ESTABLISHED	valid FAS service primitive && Data = not-allowed, unknown or faulty FMS PDU => <pre> ABT.req{ AbortIdentifier := FMS, ReasonCode := FMS-PDU-error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FMS-PDU-error }, reset VCR </pre>	CONNECTION-NOT-ESTABLISHED
S54	CONNECTION-ESTABLISHED	ConfirmedService.rsp && InvokeID (server) existent && services in .rsp and .ind are identical && PDU length ≤ Max FMS PDU sending => <pre> DTC.rsp{ Data := Confirmed-ResponsePDU }, OSCR := OSCR – 1 </pre>	CONNECTION-ESTABLISHED
S55	CONNECTION-ESTABLISHED	ConfirmedService.rsp && InvokeID (server) not existent => <pre> ABT.req{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-response }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-response }, reset VCR </pre>	CONNECTION-NOT-ESTABLISHED
S56	CONNECTION-ESTABLISHED	ConfirmedService.rsp && InvokeID (server) existent && services in .rsp and .ind are not identical => <pre> ABT.req{ AbortIdentifier := FMS, ReasonCode := service-error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := service-error }, reset VCR </pre>	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S57	CONNECTION-ESTABLISHED	ConfirmedService.rsp && InvokeID (server) existent && services in .rsp and .ind are identical && PDU length > Max FMS PDU sending => DTC.rsp{ Data := Reject-PDU{ Original InvokeID := InvokeID in ConfirmedService.req, RejectCode := PDU-size } }, OSCR := OSCR – 1	CONNECTION-ESTABLISHED
S58	CONNECTION-ESTABLISHED	DTC.cnf && Data = Confirmed-ResponsePDU && PDU length ≤ Max FMS PDU receiving && InvokeID (client) existent && services in .cnf and .req are identical => ConfirmedService.cnf{ }, OSCR := OSCR – 1	CONNECTION-ESTABLISHED
S59	CONNECTION-ESTABLISHED	DTC.cnf && Data = Confirmed-ResponsePDU && PDU length > Max FMS PDU receiving => ABT.req{ AbortIdentifier := FMS, ReasonCode := PDU-size }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := PDU-size }, reset VCR	CONNECTION-NOT-ESTABLISHED
S60	CONNECTION-ESTABLISHED	DTC.cnf && Data = Confirmed-ResponsePDU && PDU length ≤ Max FMS PDU receiving && InvokeID (client) not existent => ABT.req{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-response }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-response }, reset VCR	CONNECTION-NOT-ESTABLISHED
S61	CONNECTION-ESTABLISHED	DTC.cnf && Data = Reject-PDU && Original InvokeID (client) existent && RejectCode:= PDU-size => Reject.ind{ DetectedHere := false, Original InvokeID := InvokeID in Reject-PDU, RejectPDUType := Confirmed-Response-PDU, RejectCode := PDU-size }, OSCS := OSCL -1	CONNECTION-ESTABLISHED
S62	CONNECTION-ESTABLISHED	DTC.cnf && Data = Reject-PDU && Original InvokeID (client) not existent => ABT.req{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-response }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := InvokeID-error-response }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current state	Event or condition => action	Next state
S63	CONNECTION-ESTABLISHED	DTC.cnf && Data = Reject-PDU && Original InvokeID (client) existent && RejectCode:<> PDU-size => ABT.req{ AbortIdentifier := FMS, ReasonCode := FMS-PDU-error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FMS-PDU-error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S64	CONNECTION-ESTABLISHED	DTC.cnf && Data = Confirmed-ResponsePDU && InvokeID (client) existent && Services in .req and .cnf are not identical => ABT.req{ AbortIdentifier := FMS, ReasonCode := Service-error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Service-error }, reset VCR	CONNECTION-NOT-ESTABLISHED

7.2.4 Primitives exchanged between FAL-User and VCR PM

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

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

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

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

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

7.2.5 Primitives exchanged between FSPM and the VCR PM

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

Table 6 – Primitives issued by VCR PM to FSPM

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

Table 7 – Primitives issued by FSPM to VCR PM

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

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

8 FAL service protocol machine (FSPM)

8.1 General

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

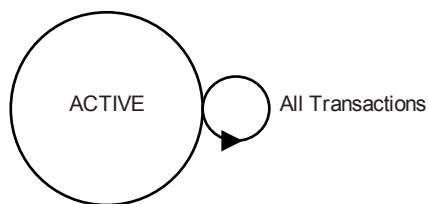


Figure 23 – State transition diagram of FSPM

8.2 FSPM state tables

Table 8 and Table 9 specify the FSPM protocol machine.

Table 8 – FSPM state table – sender transactions

#	Current state	Event or condition => action	Next state
S1	ACTIVE	ASC.req && SelectArep (Arep_Id) = "True" => ASC_req { user_data := Data, remote_dlcep_address := Remote_DLCEP_Address }	ACTIVE
S2	ACTIVE	ASC.rsp(+) && SelectArep (Arep_Id) = "True" => ASC_rsp(+) { user_data := Data }	ACTIVE
S3	ACTIVE	ASC.rsp(-) && SelectArep (Arep_Id) = "True" => ASC_rsp(-) { user_data := Data }	ACTIVE
S4	ACTIVE	DTU.req && SelectArep (Arep_Id) = "True" => DTU_req { remote_dlsap_address := Remote_DLSAP_Address, user_data := Data }	ACTIVE
S5	ACTIVE	DTC.req && SelectArep (Arep_Id) = "True" => DTC_req { user_data := Data }	ACTIVE
S6	ACTIVE	DTC.rsp && SelectArep (Arep_Id) = "True" => DTC_rsp { user_data := Data }	ACTIVE
S7	ACTIVE	Abort.req && SelectArep (Arep_Id) = "True" => Abort_req { identifier := Identifier, reason_code := Reason_Code, additional_detail := Additional_Detail }	ACTIVE
S8	ACTIVE	FCMP.req && SelectArep (Arep_Id) = "True" => FCMP_req { }	ACTIVE
S9	ACTIVE	GBM.req && SelectArep (Arep_Id) = "True" => GBM_req { }	ACTIVE

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

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

Table 9 – FSPM state table – receiver transactions

#	Current state	Event or condition => action	Next state
R1	ACTIVE	ASC_ind => ASC.ind { Arep_Id := arep_id, Data := user_data }	ACTIVE
R2	ACTIVE	ASC_cnf(+) => ASC.cnf(+) { Arep_Id := arep_id, Data := user_data }	ACTIVE
R3	ACTIVE	ASC_cnf(-) => ASC.cnf(-) { Arep_Id := arep_id, Data := user_data }	ACTIVE
R4	ACTIVE	DTU_ind => DTU.ind { Arep_Id := arep_id, Remote_DLSAP_Address := remote_dlsap_address, Duplicate_FAS-SDU := duplicate_fas_sdu, Data := user_data, Local_Timeliness := local_timeliness, Remote_Timeliness := remote_timeliness }	ACTIVE
R5	ACTIVE	DTC_ind => DTC.ind { Arep_Id := arep_id, Data := user_data }	ACTIVE
R6	ACTIVE	DTC_cnf => DTC.cnf { Arep_Id := arep_id, Data := user_data }	ACTIVE
R7	ACTIVE	Abort_ind => Abort.ind { Arep_Id := arep_id, Locally_Generated := locally_generated, Identifier := identifier, Reason_Code := reason_code, Additional_Detail := additional_detail }	ACTIVE
R8	ACTIVE	FCMP_cnf => FCMP.cnf { Arep_Id := arep_id, Status := status }	ACTIVE
R9	ACTIVE	GBM_cnf(+) => GBM.cnf(+) { Arep_Id := arep_id, Duplicate_FAS-SDU := duplicate_fas_sdu, Data := user_data, Local_Timeliness := local_timeliness, Remote_Timeliness := remote_timeliness }	ACTIVE
R10	ACTIVE	GBM_cnf(-) => GBM.cnf(-) { Arep_Id := arep_id, }	ACTIVE

#	Current state	Event or condition => action	Next state
R11	ACTIVE	<pre>FSTS_ind => FSTS.ind { Arep_Id := arep_id, Reported_Status := reported_status }</pre>	ACTIVE

8.3 Functions used by FSPM

Table 10 defines the function used by the FSPM.

Table 10 – Function SelectArep()

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

8.4 Parameters of FSPM/ARPM primitives

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

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

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

9 Application relationship protocol machines (ARPMs)

9.1 AREP mapping to data-link layer

9.1.1 General

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

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

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

9.1.2 DLL mapping of QUU AREP class

9.1.2.1 QUU AREP class formal model

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

CLASS:	QuuCI
PARENT CLASS:	QueuedUser-TriggeredUnidirectionalAREP
ATTRIBUTES:	
1.	(m) KeyAttribute: LocalDisapAddress
2.	(m) Attribute: RemoteDisapAddress
3.	(m) Attribute: LocalDisapRole (Basic, Group)
4.	(c) Constraint: Role = Source
4.1	(m) Attribute: DefaultQosAsSender
4.2	(m) Attribute: DIIPriority (Urgent, Normal, TimeAvailable)
4.3	(m) Attribute: MaxConfirmDelayOnUnitdata
4.4	(m) Attribute: DIpduAuthentication (Ordinary, Source, Maximal)
4.5	(m) Attribute: DISchedulingPolicy (Implicit)
4.6	(m) Attribute: ExplicitQueue (True, False)
5.	(c) Constraint: ExplicitQueue = True
5.1	(m) Attribute: QueueBindings—either for a sender or a receiver
5.2	(m) Attribute: QueueIdentifier
5.3	(m) Attribute: MaxQueueDepth
5.4	(m) Attribute: MaxDisduSize
DLL SERVICES:	
1.	(m) OpsService: DL-Unitdata
2.	(c) Constraint: ExplicitQueue = True
3.	(m) OpsService: DL-Get

9.1.2.2 Attributes

9.1.2.2.1 LocalDisapAddress

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

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

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

9.1.2.2.2 RemoteDisapAddress

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

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

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

9.1.2.2.3 LocalDisapRole

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

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

9.1.2.2.4 DefaultQosAsSender

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

9.1.2.2.5 DllPriority

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

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

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

9.1.2.2.6 MaxConfirmDelayOnUnitdata

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

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

9.1.2.2.7 DlpduAuthentication

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

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

9.1.2.2.8 DISchedulingPolicy

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

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

9.1.2.2.9 ExplicitQueue

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

9.1.2.2.10 QueueBindings

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

9.1.2.2.11 QueueIdentifier

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

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

9.1.2.2.12 MaxQueueDepth

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

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

9.1.2.2.13 MaxDlsduSize

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

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

9.1.2.3 DLL services

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

9.1.3 DLL mapping of QUB AREP class

9.1.3.1 QUB AREP class formal model

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

CLASS: QubCo

PARENT CLASS: QueuedUser-TriggeredBidirectionalAREP

ATTRIBUTES:

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

DLL SERVICES:

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

9.1.3.2 Attributes**9.1.3.2.1 LocalDlcepAddress**

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

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

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

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

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

9.1.3.2.2 RemoteDlcepAddress

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

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

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

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

9.1.3.2.3 DlcepDlIdentifier

The DlcepDlIdentifier attribute identifies the DLCEP.

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

9.1.3.2.4 DlcepRole

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

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

9.1.3.2.5 QosParameterSet

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

9.1.3.2.6 DlcepClass

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

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

9.1.3.2.7 DlcepDataDeliveryFeatures

These two attributes specify data delivery features of the DLL.

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

The FromRequesterToResponder and FromResponderToRequester attributes shall have the same value.

9.1.3.2.8 FromRequesterToResponder

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

9.1.3.2.9 FromResponderToRequester

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

9.1.3.2.10 DllPriority

This attribute specifies the configured value of the DLL priority.

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

9.1.3.2.11 DllPriorityNegotiated

This attribute specifies the negotiated value of the DLL priority.

9.1.3.2.12 DlpduAuthentication

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

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

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

9.1.3.2.13 ResidualActivityAsReceiver

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

9.1.3.2.14 MaxConfirmDelay

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

9.1.3.2.15 MaxConfirmDelayOnDIConnect

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

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

9.1.3.2.16 MaxConfirmDelayOnDIData

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

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

9.1.3.2.17 DISchedulingPolicy

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

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

9.1.3.2.18 ExplicitQueue**9.1.3.2.19 MaxDlsduSizeFromRequester**

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

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

9.1.3.2.20 MaxDlsduSizeFromResponder

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

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

9.1.3.2.21 MaxDlsduSizeFromRequesterNegotiated

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

9.1.3.2.22 MaxDlsduSizeFromResponderNegotiated

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

9.1.3.2.23 SendingBufferOrQueueIdentifier

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

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

9.1.3.2.24 ReceivingBufferOrQueueIdentifier

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

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

9.1.3.2.25 MaxSendingQueueDepth

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

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

9.1.3.2.26 MaxReceivingQueueDepth

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

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

9.1.3.3 DLL services

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

9.1.4 DLL mapping of BNU AREP class

9.1.4.1 BNU AREP class formal model

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

CLASS: BnuCo

PARENT CLASS: BufferedNetworkScheduledUnidirectionalAREP

ATTRIBUTES:

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

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

DLL SERVICES:

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

9.1.4.2 Attributes**9.1.4.2.1 PublisherDlcepAddress**

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

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

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

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

9.1.4.2.2 DlcepDlIdentifier**9.1.4.2.3 DisapRole****9.1.4.2.4 QosParameterSet****9.1.4.2.5 DlcepClass****9.1.4.2.6 DlIPriority**

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

9.1.4.2.7 DlPduAuthentication**9.1.4.2.8 ResidualActivity****9.1.4.2.9 MaxConfirmDelayOnDIConnect****9.1.4.2.10 DISchedulingPolicy**

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

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

9.1.4.2.11 FromRequesterToResponder

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

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

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

9.1.4.2.12 MaxDisduSizeFromRequester

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

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

9.1.4.2.13 SendingBufferOrQueueIdentifier

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

9.1.4.2.14 FromResponderToRequester

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

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

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

9.1.4.2.15 MaxDisduSizeFromResponder

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

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

9.1.4.2.16 ReceivingBufferOrQueueIdentifier

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

9.1.4.2.17 SenderTimeliness

9.1.4.2.18 PublisherDITimelinessClass

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

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

9.1.4.2.19 PublisherTimeWindowSize

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

9.1.4.2.20 PublisherSynchronizingDLcep

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

9.1.4.2.21 ReceiverTimeliness

9.1.4.2.22 SubscriberDITimelinessClass

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

9.1.4.2.23 SubscriberTimeWindowSize

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

9.1.4.2.24 SubscriberSynchronizingDLcep

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

9.1.4.2.25 LasScheduled

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

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

9.1.4.3 DLL services

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

9.2 Application relationship protocol machines (ARPMs)

9.2.1 AR protocol machine (ARPM) for QUU AREP

9.2.1.1 QUU ARPM states

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

Table 12 – QUU ARPM states

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

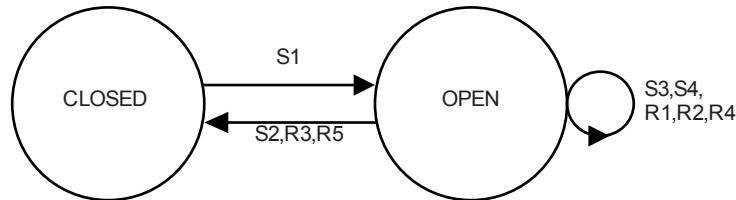


Figure 24 – State transition diagram of the QUU ARPM

9.2.1.2 QUU ARPM state table

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

Table 13 – QUU ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state
S1	CLOSED	ASC_req => ASC_cnf(+) { arep_id := GetArepId (), user_data := "null" }	OPEN
S2	OPEN	Abort_req => (no actions taken)	CLOSED
S3	OPEN	DTU_req && ConfigurationType = "Linked" && Role = "Source" => FAS-PDU_req { dmpm_service_name := "DMPM_Unitdata_req", arep_id := GetArepId (), dlsdu := BuildFAS-PDU (fas_pdu_name := "DTU_PDU", fas_data := user_data) }	OPEN
S4	OPEN	DTU_req && ConfigurationType = "Free" && Role = "Source" => RemoteDlsapAddress := remote_dlsap_address, FAS-PDU_req { dmpm_service_name := "DMPM_Unitdata_req", arep_id := GetArepId (), dlsdu := BuildFAS-PDU (fas_pdu_name := "DTU_PDU", fas_data := user_data) }	OPEN

Table 14 – QUB ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state
R1	OPEN	FAS-PDU_ind && dmpm_service_name = "DMPM_Unitdata_ind" && ConfigurationType = "Linked" && RemoteDisapAddress = calling_address && Role = "Sink" && FAS_Pdu_Type (fas_pdu) = "DTU_PDU" => DTU_ind { arep_id := GetArepId (), user_data := fas_pdu }	OPEN
R2	OPEN	FAS-PDU_ind && dmpm_service_name = "DMPM_Unitdata_ind" && ConfigurationType = "Free" && Role = "Sink" && FAS_Pdu_Type (fas_pdu) = "DTU_PDU" => DTU_ind { arep_id := GetArepId (), remote_disap_address := calling_address, user_data := fas_pdu }	OPEN
R3	OPEN	FAS-PDU_ind && dmpm_service_name = "DMPM_Unitdata_ind" && Role = "Sink" && FAS_Pdu_Type (fas_pdu) <> "DTU_PDU" => Abort_ind { arep_id := GetArepId (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" }	CLOSED
R4	OPEN	ErrorToARPM => (no actions taken)	OPEN
R5	OPEN	FAS-PDU_ind && Role = "Source" => Abort_ind { arep_id := GetArepId (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid Event for Role", additional_detail := "null" }	CLOSED

9.2.2 AR protocol machine (ARPM) for QUB AREP

9.2.2.1 QUB ARPM states

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

Table 15 – QUB ARPM states

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

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

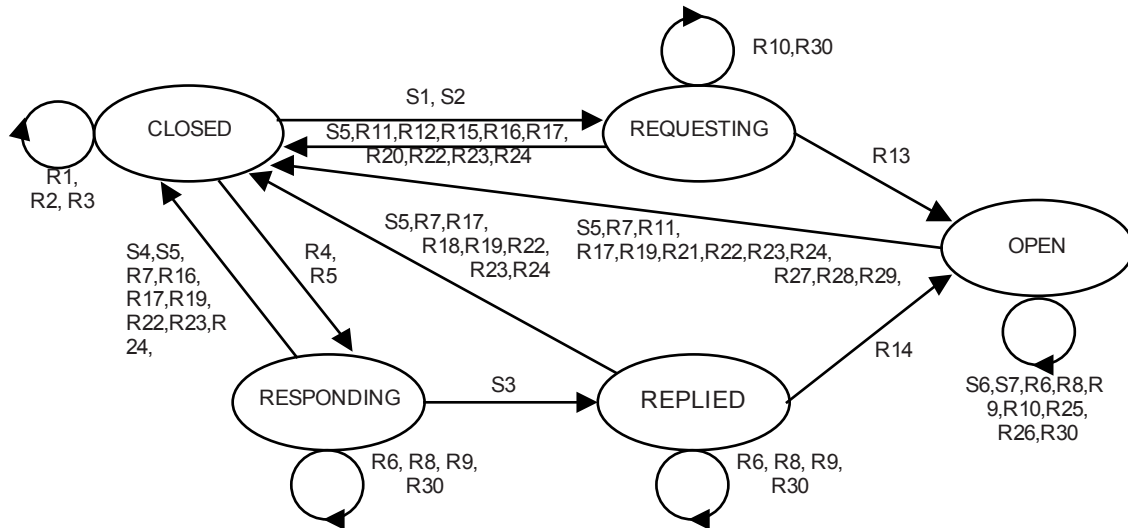


Figure 25 – State transition diagram of QUB ARPM

9.2.2.2 QUB ARPM state table

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

Table 16 – QUB ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state
S1	CLOSED	ASC_req && Initiator = "True" && RemoteAddressConfigurationType := "Free" RemoteDlcepAddress := remote_dlcep_address, => FAS-PDU_req { dmpm_service_name := "DMPM_Connect_req", arep_id := GetArepld (), called_address := "default dlsap address", calling_address := "default dlsap address", local_dlcep_address := LocalDlcep, dlsdu := BuildFAS-PDU (fas_pdu_name := "ASC_ReqPDU", calling_dlcep_address := LocalDlcepAddress, called_dlcep_address := RemoteDlcepAddress, fas_data := user_data) } }	REQ
S2	CLOSED	ASC_req && Initiator = "True" && RemoteAddressConfigurationType := "Linked" => FAS-PDU_req { dmpm_service_name := "DMPM_Connect_req", arep_id := GetArepld (), called_address := "default dlsap address", calling_address := "default dlsap address", local_dlcep_address := LocalDlcep, dlsdu := BuildFAS-PDU (fas_pdu_name := "ASC_ReqPDU", calling_dlcep_address := LocalDlcepAddress, called_dlcep_address := RemoteDlcepAddress, fas_data := user_data) } }	REQ

#	Current state	Event or condition => action	Next state
S3	RSP	<pre> ASC_rsp(+) => FAS-PDU_req { dmpm_service_name := "DMPM_Connect_rsp", arep_id := GetArepld (), responding_address := "default dlsap address", local_dlcep_address := LocalDlcep, dlcep_dl_id := DlcepDlIdentifier, dlsdu := BuildFAS-PDU (fas_pdu_name := "ASC_RspPDU", fas_data := user_data) } </pre>	REPL
S4	RSP	<pre> ASC_rsp(-) => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "connection rejection-transient condition", dlsdu := BuildFAS-PDU (fas_pdu_name := "ASC_ErrPDU", fas_data := user_data) } </pre>	CLOSED
S5	NOT CLOSED	<pre> Abort_req => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "disconnection-normal condition", dlsdu := BuildFAS-PDU (fas_pdu_name := "Abort_PDU", fas_id := identifier, fas_reason_code := reason_code, fas_additional_detail := additional_detail) } </pre>	CLOSED
S6	OPEN	<pre> DTC_req && Role = "Client" "Peer" => FAS-PDU_req { dmpm_service_name := "DMPM_Data_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, dlsdu := BuildFAS-PDU (fas_pdu_name := "DTC_ReqPDU", fas_data := user_data) } </pre>	OPEN
S7	OPEN	<pre> DTC_rsp && Role = "Server" "Peer" => FAS-PDU_req { dmpm_service_name := "DMPM_Data_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, dlsdu := BuildFAS-PDU (fas_pdu_name := "DTC_RspPDU", fas_data := user_data) } </pre>	OPEN

Table 17 – QUB ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state
R1	CLOSED	<pre> Connect_ind && Initiator = "True" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Multiple Initiators", dlsdu := "null" } </pre>	CLOSED
R2	CLOSED	<pre> Connect_ind && Initiator = "False" && FAS_Pdu_Type (dls_user_data) <> "ASC_ReqPDU" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Invalid FAS-PDU", dlsdu := "null" } </pre>	CLOSED
R3	CLOSED	<pre> Connect_ind && Initiator = "False" && FAS_Pdu_Type (dls_user_data) = "ASC_ReqPDU" && RemoteAddressConfigurationType = "Linked" && RemoteDlcepAddress <> calling_dlcep_address => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Remote Address Mismatch", dlsdu := "null" } </pre>	CLOSED
R4	CLOSED	<pre> Connect_ind && Initiator = "False" && FAS_Pdu_Type (dls_user_data) = "ASC_ReqPDU" && RemoteAddressConfigurationType = "Free" => RemoteDlcepAddress := calling_dlcep_address, DlcepDlIdentifier := dlcep_dl_id, MaxDlsduSizeFromRequesterNegotiated := dlsdu_size_from_requester, MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder, ASC_ind { arep_id := GetArepld (), user_data := dls_user_data } </pre>	RSP
R5	CLOSED	<pre> Connect_ind && Initiator = "False" && FAS_Pdu_Type (dls_user_data) = "ASC_ReqPDU" && RemoteAddressConfigurationType = "Linked" && RemoteDlcepAddress = calling_dlcep_address => DlcepDlIdentifier := dlcep_dl_id, MaxDlsduSizeFromRequesterNegotiated := dlsdu_size_from_requester, MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder, ASC_ind { arep_id := GetArepld (), user_data := dls_user_data } </pre>	RSP

#	Current state	Event or condition => action	Next state
R6	RSP REPL OPEN	<pre> Connect_ind && Initiator = "False" && RemoteAddressConfigurationType = "Linked" && RemoteDlcepAddress <> calling_dlcep_address => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Remote Address Mismatch", dlsdu := "null" } </pre>	SAME
R7	RSP REPL OPEN	<pre> Connect_ind && Initiator = "False" && RemoteDlcepAddress = calling_dlcep_address => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Invalid FAS-PDU", dlsdu := "null" }, Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU" } </pre>	CLOSED
R8	RSP REPL OPEN	<pre> Connect_ind && Initiator = "False" && FAS_Pdu_Type (dls_user_data) = "ASC_ReqPDU" && RemoteAddressConfigurationType = "Free" && RemoteDlcepAddress <> calling_dlcep_address => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "AREP Busy", dlsdu := "null" } </pre>	SAME
R9	RSP REPL OPEN	<pre> Connect_ind && Initiator = "False" && FAS_Pdu_Type (dls_user_data) <> "ASC_ReqPDU" && RemoteAddressConfigurationType = "Free" && RemoteDlcepAddress <> calling_dlcep_address => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Invalid FAS-PDU", dlsdu := "null" } </pre>	SAME
R10	REQ OPEN	<pre> Connect_ind && Initiator = "True" && RemoteDlcepAddress <> calling_dlcep_address => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Multiple Initiators", dlsdu := "null" } </pre>	SAME

#	Current state	Event or condition => action	Next state
R11	REQ OPEN	<pre> Connect_ind && Initiator = "True" && RemoteDlcepAddress = calling_dlcep_address => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := dlcep_dl_id (from Connect_ind), reason := "Multiple Initiators", dlsdu := "null" }, Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Multiple Initiators" } </pre>	CLOSED
R12	REQ	<pre> Connect_cnf && FAS_Pdu_Type (dls_user_data) <> "ASC_RspPDU" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDllIdentifier, reason := "Invalid FAS-PDU", dlsdu := "null" }, Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU" } </pre>	CLOSED
R13	REQ	<pre> Connect_cnf && FAS_Pdu_Type (dls_user_data) = "ASC_RspPDU" => MaxDlsduSizeFromRequesterNegotiated := dlsdu_size_from_requester, MaxDlsduSizeFromResponderNegotiated := dlsdu_size_from_responder, DllPriorityNegotiated := dll_priority, ASC_cnf(+) { arep_id := GetArepld (), user_data := dls_user_data } </pre>	OPEN
R14	REPL	<pre> FAS-PDU_ind && dmpm_service_name = "DMPM_Connection_Established_ind" => (no actions taken) </pre>	OPEN
R15	REQ	<pre> FAS-PDU_ind && dmpm_service_name = "DMPM_Disconnect_ind" && FAS_Pdu_Type (fas_pdu) = "ASC_ErrPDU" => ASC_cnf(-) { arep_id := GetArepld (), user_data := dls_user_data } </pre>	CLOSED

#	Current state	Event or condition => action	Next state
R16	REQ RSP	<pre> FAS-PDU_ind && dmpm_service_name <> "DMPM_Disconnect_ind" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "Invalid DL-Event", dlsdu := "null" }, Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid DL-Event" } </pre>	CLOSED
R17	NOT CLOSED	<pre> FAS-PDU_ind && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu <> "null" && FAS_Pdu_Type (fas_pdu) = "Abort_PDU" => Abort_ind{ arep_id := GetArepld (), locally_generated := "False", identifier := AbortIdentifier (fas_pdu), reason_code := AbortReason (fas_pdu), additional_detail := AbortDetail (fas_pdu) } </pre>	CLOSED
R18	REPL	<pre> FAS-PDU_ind && ((dmpm_service_name <> "DMPM_Disconnect_ind") && (dmpm_service_name <>"DM_Connection_Established_ind")) => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "Invalid DL-Event", dlsdu := "null" }, Abort_ind{ arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid DL-Event", additional_detail := "null" } </pre>	CLOSED
R19	REPL RSP OPEN	<pre> FAS-PDU_ind && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu <> "null" && FAS_Pdu_Type (fas_pdu) <> "Abort_PDU" => Abort_ind{ arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" } </pre>	CLOSED

#	Current state	Event or condition => action	Next state
R20	REQ	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu <> "null" && ((FAS_Pdu_Type (fas_pdu) <> "Abort_PDU") && (FAS_Pdu_Type (fas_pdu) <> "ASC_ErrPDU")) => Abort_ind{ arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" } }</pre>	CLOSED
R21	OPEN	<pre>FAS-PDU_ind && ((dmpm_service_name <> "DMPM_Disconnect_ind") && (dmpm_service_name <> "DMPM_Data_ind")) => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "Invalid DL-Event", dlsdu := "null" }, Abort_ind{ arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid DL-Event", additional_detail := "null" } }</pre>	CLOSED
R22	NOT CLOSED	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu = "null" && originator = "remote_dls_provider" => Abort_ind{ arep_id := GetArepld (), locally_generated := "False", identifier := "Data-link layer", reason_code := reason, additional_detail := "null" } }</pre>	CLOSED
R23	NOT CLOSED	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu = "null" && originator = "remote_dls_user" => Abort_ind{ arep_id := GetArepld (), locally_generated := "False", identifier := "FAS", reason_code := reason, additional_detail := "null" } }</pre>	CLOSED
R24	NOT CLOSED	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu = "null" && originator = "local_dls_provider" => Abort_ind{ arep_id := GetArepld (), locally_generated := "True", identifier := "Data-link layer", reason_code := reason, additional_detail := "null" } }</pre>	CLOSED

#	Current state	Event or condition => action	Next state
R25	OPEN	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Data_ind" && Role = "Peer" "Server" && FAS_Pdu_Type (fas_pdu) = "DTC_ReqPDU" => DTC_ind { arep_id := GetArepld (), user_data := fas_pdu } }</pre>	OPEN
R26	OPEN	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Data_ind" && Role = "Client" "Peer" && FAS_Pdu_Type (fas_pdu) = "DTC_RspPDU" => DTC_cnf { arep_id := GetArepld (), user_data := fas_pdu } }</pre>	OPEN
R27	OPEN	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Data_ind" && Role = "Server" && FAS_Pdu_Type (fas_pdu) <> "DTC_ReqPDU" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "Invalid FAS-PDU", dlsdu := "null" }, Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" } }</pre>	CLOSED
R28	OPEN	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Data_ind" && Role = "Client" && FAS_Pdu_Type (fas_pdu) <> "DTC_RspPDU" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "Invalid FAS-PDU", dlsdu := "null" }, Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" } }</pre>	CLOSED

#	Current state	Event or condition => action	Next state
R29	OPEN	<pre> FAS-PDU_ind && Role = "Peer" && dmpm_service_name = "DMPM_Data_ind" && ((FAS_Pdu_Type (fas_pdu) <> "DTC_ReqPDU") && (FAS_Pdu_Type (fas_pdu) <> "DTC_RspPDU")) => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDIIdentifier, reason := "Invalid FAS-PDU", dlsdu := "null" }, Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" } </pre>	CLOSED
R30	NOT CLOSED	<pre> ErrorToARPM => (no actions taken) </pre>	SAME

NOTE It is a local matter to report an error status to network management entities. The ARPM does not abort the existing connections. The FAS user may issue an Abort request to disconnect the current connection, depending on the type of status information conveyed by the ErrorToARPM primitive.

9.2.3 AR protocol machine (ARPM) for BNU AREP

9.2.3.1 BNU ARPM states

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

Table 18 – BNU ARPM states

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

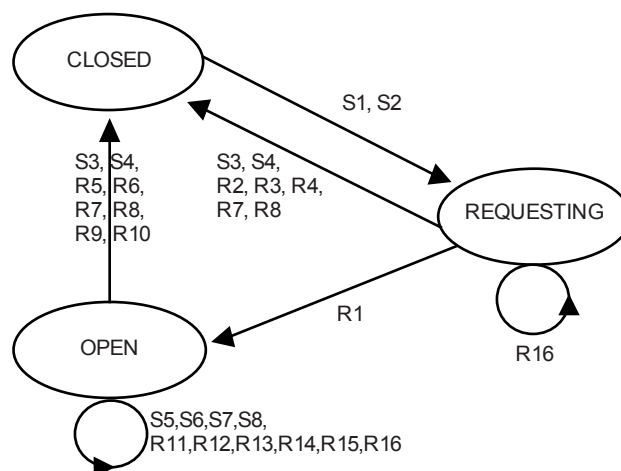


Figure 26 – State transition diagram of the BNU ARPM

9.2.3.2 BNU ARPM state table

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

Table 19 – BNU ARPM state table – sender transactions

#	Current state	Event or condition => action	Next state
S1	CLOSED	<pre> ASC_req && Role = "Publisher" => FAS-PDU_req { dmpm_service_name := "DMPM_Connect_req", arep_id := GetArepld (), called_address := "null", calling_address := "default dlsap address", local_dlcep_address := PublisherDlcepAddress, dlsdu := "null" } </pre>	REQ
S2	CLOSED	<pre> ASC_req && Role = "Subscriber" => FAS-PDU_req { dmpm_service_name := "DMPM_Connect_req", arep_id := GetArepld (), called_address := PublisherDlcepAddress, calling_address := "default dlsap address", local_dlcep_address := "default subscriber dlcep address", dlsdu := "null" } </pre>	REQ
S3	NOT CLOSED	<pre> Abort_req && Role = "Publisher" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "disconnection-normal condition", dlsdu := BuildFAS-PDU (fas_pdu_name := "Abort_PDU", fas_id := identifier, fas_reason_code := reason_code, fas_additional_detail := additional_detail) } </pre>	CLOSED
S4	NOT CLOSED	<pre> Abort_req && Role = "Subscriber" => FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "disconnection-normal condition", dlsdu := "null" } </pre>	CLOSED
S5	OPEN	<pre> DTU_req && Role = "Publisher" => FAS-PDU_req { dmpm_service_name := "DMPM_Put_req", arep_id := GetArepld (), dlsdu := BuildFAS-PDU (fas_pdu_name := "DTU_PDU", fas_sdu := user_data) } </pre>	OPEN
S6	OPEN	<pre> FCMP_req && Role = "Publisher" => FAS-PDU_req { dmpm_service_name := "DMPM_Compel_Service_req", arep_id := GetArepld (), action_class := "Local" } </pre>	OPEN

#	Current state	Event or condition => action	Next state
S7	OPEN	FCMP_req && Role = "Subscriber" => FAS-PDU_req { dmpm_service_name := "DMPM_Compel_Service_req", arep_id := GetArepld (), action_class := "Remote" }	OPEN
S8	OPEN	GBM_req && Role = "Subscriber" => FAS-PDU_req { dmpm_service_name := "DMPM_Get_req", arep_id := GetArepld () }	OPEN

Table 20 – BNU ARPM state table – receiver transactions

#	Current state	Event or condition => action	Next state
R1	REQ	Connect_cnf => ASC_cnf(+) { arep_id := GetArepld (), user_data := "null" }	OPEN
R2	REQ	FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu = "null" && originator = "local_dls_provider" => ASC_cnf(-) { arep_id := GetArepld (), user_data := "null" }	CLOSED
R3	REQ	FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu = "null" && originator = "remote_dls_provider" => ASC_cnf(-) { arep_id := GetArepld (), user_data := "null" }	CLOSED
R4	REQ	FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name <> "DMPM_Disconnect_ind" => Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid DI Event", additional_detail := "null" }	CLOSED
R5	OPEN	FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu = "null" && originator = "local_dls_provider" => Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "Data-link layer", reason_code := reason, additional_detail := "null" }	CLOSED

#	Current state	Event or condition => action	Next state
R6	OPEN	<pre> FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Disconnect_ind" && dlsdu = "null" && originator = "remote_dls_provider" => Abort_ind { arep_id := GetArepld (), locally_generated := "False", identifier := "Data-link layer," reason_code := reason, additional_detail := "null" } </pre>	CLOSED
R7	NOT CLOSED	<pre> FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Disconnect_ind" && Fas_Pdu_Type (fas_pdu) = "Abort_PDU" => Abort_ind { arep_id := GetArepld (), locally_generated := "False", identifier := AbortIdentifier (fas_pdu), reason_code := AbortReason (fas_pdu), additional_detail := AbortDetail (fas_pdu) } </pre>	CLOSED
R8	NOT CLOSED	<pre> FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Disconnect_ind" && fas_pdu <> "null" && Fas_Pdu_Type (fas_pdu) <> "Abort_PDU" => Abort_ind { arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" } </pre>	CLOSED
R9	OPEN	<pre> FAS-PDU_ind && Role = "Subscriber" && ((dmpm_service_name <> "DMPM_Buffer_Received_ind") && (dmpm_service_name <> "DMPM_Disconnect_ind") && (dmpm_service_name <> "DMPM_Compel_Service_cnf") && (dmpm_service_name <> "DMPM_Get_cnf") && (dmpm_service_name <> "DMPM_Buffer_Sent_ind")) => Abort_ind{ arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid DI Event", additional_detail := "null" }, FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDIIdentifier, reason := "Invalid DL-Event", dlsdu := "null" } </pre>	CLOSED

#	Current state	Event or condition => action	Next state
R10	OPEN	<pre>FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Buffer_Received_ind" && FAS_pdu_type <> "DTU_PDU" => Abort_ind{ arep_id := GetArepld (), locally_generated := "True", identifier := "FAS", reason_code := "Invalid FAS-PDU", additional_detail := "null" }, FAS-PDU_req { dmpm_service_name := "DMPM_Disconnect_req", arep_id := GetArepld (), dlcep_dl_id := DlcepDlIdentifier, reason := "Invalid FAS-PDU", dlsdu := "null" } }</pre>	CLOSED
R11	OPEN	<pre>FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Buffer_Received_ind" && FAS_Pdu_Type (fas_pdu) = "DTU_PDU" => DTU_ind { arep_id := GetArepld (), duplicate_fas_sdu := duplicate_dlsdu, user_data := fas_pdu, local_timeliness := local_dle_timeliness, remote_timeliness := remote_dle_timeliness } }</pre>	OPEN
R12	OPEN	<pre>FAS-PDU_ind && dmpm_service_name = "DMPM_Compel_Service_cnf" => FCPM_cnf { status := dl_status } }</pre>	OPEN
R13	OPEN	<pre>FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Get_cnf" && status = "success" => GBM_cnf(+) { arep_id := GetArepld (), duplicate_fas_sdu := duplicate_dlsdu, user_data := fas_pdu, local_timeliness := local_dle_timeliness, remote_timeliness := remote_dle_timeliness } }</pre>	OPEN
R14	OPEN	<pre>FAS-PDU_ind && Role = "Subscriber" && dmpm_service_name = "DMPM_Get_cnf" && status <> "success" => GBM_cnf(-) { arep_id := GetArepld () } }</pre>	OPEN
R15	OPEN	<pre>FAS-PDU_ind && Role = "Publisher" && dmpm_service_name = "DMPM_Buffer_Sent_ind" => FSTS_ind { arep_id := GetArepld (), reported_status := "Buffer-Sent" } }</pre>	OPEN
R16	NOT CLOSED	<pre>ErrorToARPM => (no actions taken)</pre>	SAME

NOTE It is a local matter to report an error status to network management entities. The ARPM does not abort the existing connections. The FAS user may issue an Abort request to disconnect the current connection, depending on the type of status information conveyed by the ErrorToARPM primitive.

9.3 AREP state machine primitive definitions

9.3.1 Primitives exchanged between DMPM and ARPM

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

Table 21 – Primitives issued from ARPM to DMPM

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

Table 22 – Primitives issued by DMPM to ARPM

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

9.3.2 Parameters of ARPM/DMPM primitives

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

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

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

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

9.4 AREP state machine functions

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

Table 24 – Function GetArepld()

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

Table 25 – Function BuildFAS-PDU

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

Table 26 – Function FAS_Pdu_Type

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

Table 27 – Function AbortIdentifier

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

Table 28 – Function AbortReason

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

Table 29 – Function AbortDetail

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

10 DLL mapping protocol machine (DMPM)

10.1 DMPM States

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

Table 30 – DMPM state descriptions

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

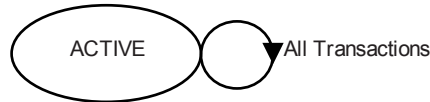


Figure 27 – State transition diagram of DMPM

10.2 DMPM state table

Table 31 and Table 32 specify the DMPM state machine.

Table 31 – DMPM state table – sender transactions

#	Current state	Event or condition => action	Next state
S1	ACTIVE	FAS-PDU_req && dmpm_service_name = "DMPM_Connect_req" => PickArep (arep_id), DL_Connect.req { calling_address := calling_address, dl_dlcep_address := local_dlcep_address, dl_dlcep_class := DlcepClass, dl_delivery_from_requester := FromRequesterToResponder, *1 dl_delivery_from_responder := FromResponderToRequester, *2 dl_dll_priority := DIIPriority, dl_max_confirm_delay_on_connect := MaxConfirmDelayOnDIConnect, dl_max_confirm_delay_on_data := MaxConfirmDelayOnDIData, *1, *2 dl_dlpdu_authentication := DlpduAuthentication, dl_residual_activity_as_sender := ResidualActivityAsSender, *1 dl_residual_activity_as_receiver := ResidualActivityAsReceiver, *1, *2 dl_scheduling_policy := DISchedulingPolicy, dl_dlsdu_size_from_requester := MaxDlsduSizeFromRequester, *1 dl_dlsdu_size_from_responder := MaxDlsduSizeFromResponder, *2 dl_buffer_queue_binding_as_sender := SendingBufferOrQueueIdentifier, *1 dl_buffer_queue_binding_as_receiver := ReceivingBufferOrQueueIdentifier, *2 dl_dls_user_data := dlsdu, *1, *2 dl_sender_dl_timeliness_class := SenderDITimelinessClass, *4 dl_sender_time_window_size := SenderTimeWindowSize, *4 dl_sender_synchronizing_dlcep := SenderSynchronizingDlcep, *4 dl_receiver_dl_timeliness_class := ReceiverDITimelinessClass, *3 dl_receiver_time_window_size := ReceiverTimeWindowSize, *3 dl_receiver_synchronizing_dlcep := ReceiverSynchronizingDlcep *3 DlcepDIIdentifier := dl_dlcep_dl_id (from DL_Connect.req(out)) } *1: Not used with BNU AREPs whose role is Subscriber. *2: Not used with BNU AREPs whose role is Publisher. *3: Only used with BNU AREPs whose role is Subscriber. *4: Only used with BNU AREPs whose role is Publisher.	ACTIVE

#	Current state	Event or condition => action	Next state
S2	ACTIVE	<pre> FAS-PDU_req && dmpm_service_name = "DMPM_Connect_rsp" => PickArep (arep_id), DL_Connect.rsp { dl_dlcep_dl_id := dlcep_dl_id, dl_responding_address := responding_address, dl_dlcep_address := local_dlcep_address, dl_dlcep_class := DlcepClass, dl_delivery_from_requester := FromRequesterToResponder, dl_delivery_from_responder := FromResponderToRequester, dl_dli_priority := DliPriority, dl_max_confirm_delay_on_connect := MaxConfirmDelayOnDIConnect, dl_max_confirm_delay_on_data := MaxConfirmDelayOnDIData, dl_dlpdu_authentication := DlpduAuthentication, dl_residual_activity_as_sender := ResidualActivityAsSender, dl_residual_activity_as_receiver := ResidualActivityAsReceiver, dl_scheduling_policy := DISchedulingPolicy, dl_dlsdu_size_from_requester := MaxDlsduSizeFromRequesterNegotiated, dl_dlsdu_size_from_responder := MaxDlsduSizeFromResponderNegotiated, dl_buffer_queue_binding_as_sender := SendingBufferOrQueueIdentifier, dl_buffer_queue_binding_as_receiver := ReceivingBufferOrQueueIdentifier, dl_dls_user_data := dlsdu } </pre>	ACTIVE
S3	ACTIVE	<pre> FAS-PDU_req && dmpm_service_name = "DMPM_Disconnect_req" => PickArep (arep_id), DL_Disconnect.req { dl_dlcep_dl_id := dlcep_dl_id, dl_reason := reason, dl_dls_user_data := dlsdu } </pre>	ACTIVE
S4	ACTIVE	<pre> FAS-PDU_req && dmpm_service_name = "DMPM_Data_req" => PickArep (arep_id), dmpm_request_id := SelectIdentifier(), DL_Data.req { dl_request_dls_user_id := dmpm_request_id, dl_dlcep_dl_id := dlcep_dl_id, dl_dls_user_data := dlsdu, } </pre>	ACTIVE
S5	ACTIVE	<pre> FAS-PDU_req && dmpm_service_name = "DMPM_Unitdata_req" => dmpm_request_id := SelectIdentifier(), PickArep (arep_id), DL_Unitdata.req { dl_request_dls_user_id := dmpm_request_id, dl_called_address := RemoteDisapAddress, dl_calling_address := LocalDisapAddress, dl_dli_priority := DliPriority, dl_max_confirm_delay := MaxConfirmDelayOnUnitdata, dl_remote_dle_confirmed := "False", dl_dls_user_data := dlsdu } </pre>	ACTIVE

#	Current state	Event or condition => action	Next state
S6	ACTIVE	<pre> FAS-PDU_req && dmpm_service_name = "DMPM_Put_req" => PickArep (arep_id), DL_Put.in { dl_buffer_id := SendingBufferOrQueueIdentifier, dl_dls_user_data := dlsdu } DL_Put.out && dl_status = "success" (no actions taken) DL_Put.out && dl_status <> "success" ErrorToARPM { originator := "local_dls", reason := dl_status } </pre>	ACTIVE
S7	ACTIVE	<pre> FAS-PDU_req && dmpm_service_name = "DMPM_Get_req" => PickArep (arep_id), DL_Get.in { dl_buffer_or_queue_id := ReceivingBufferOrQueueIdentifier } DL_Get.out && dl_reported_service_identification_class = "NONE" FAS-PDU_ind { dmpm_service_name := "DMPM_Get_cnf", status := dl_status, duplicate_dlsdu := dmpm_duplicate_dlsdu, fas_pdu := dl_dls_user_data, local_dle_timeliness := dl_local_dle_timeliness, remote_dle_timeliness := dl_sender_and_remote_dle_timeliness } </pre>	ACTIVE
S8	ACTIVE	<pre> FAS-PDU_req && dmpm_service_name = "DMPM_Compel_req" => PickArep (arep_id), DL_Compel_Service.in { dl_action_class := action_class, dl_dlcep_dl_id := DlcepDlIdentifier } DL_Compel_Service.out FAS-PDU_ind { dmpm_service_name := "DMPM_Compel_Service_cnf", status := dl_status } </pre>	ACTIVE

Table 32 – DMPM state table – receiver transactions

#	Current state	Event or condition => action	Next state
R1	ACTIVE	DL_Connect.ind && LocateQubArep (dl_dls_user_data) = "NOT FOUND" => DL_Disconnect.req { dl_dlcep_dl_id := dl_dlcep_dl_id (from indication primitive), dl_reason := "AREP Not Found", dl_dls_user_data := "null" }	ACTIVE
R2	ACTIVE	DL_Connect.ind && LocateQubArep (dl_dls_user_data) = "NOT QUB" => DL_Disconnect.req { dl_dlcep_dl_id := dl_dlcep_dl_id (from indication primitive), dl_reason := "Invalid AREP type", dl_dls_user_data := "null" }	ACTIVE
R3	ACTIVE	DL_Connect.ind && LocateQubArep (dl_dls_user_data) = "True" => Connect_ind { dlcep_dl_id := dl_dlcep_dl_id, calling_dlcep_address := RequestingAREP, called_dlcep_address := RespondingAREP, called_address := dl_called_address, calling_address := dl_calling_address, dlcep_class := dl_dlcep_class, delivery_from_requester := dl_delivery_from_requester, delivery_from_responder := dl_delivery_from_responder, dll_priority := dl_dll_priority, max_confirm_delay_on_connect := dl_max_confirm_delay_on_connect, max_confirm_delay_on_data := dl_max_confirm_delay_on_data, dlsdu_size_from_requester := dl_dlsdu_size_from_requester, dlsdu_size_from_responder := dl_dlsdu_size_from_responder, dls_user_data := dl_dls_user_data }	ACTIVE
R4	ACTIVE	DL_Connect.cnf && FindAREP (dl_dlcep_dl_id) = "False" => DL_Disconnect.req { dl_dlcep_dl_id := dl_dlcep_dl_id (from confirmation primitive), dl_reason := "DLCEP Not Found", dl_dls_user_data := "null" }	ACTIVE
R5	ACTIVE	DL_Connect.cnf && FindAREP (dl_dlcep_dl_id) = "True" => Connect_cnf { responding_address := dl_responding_address, dlcep_class := dl_dlcep_class, delivery_from_requester := dl_delivery_from_requester, *1 delivery_from_responder := dl_delivery_from_responder, *2 dll_priority := dl_dll_priority, max_confirm_delay_on_connect := dl_max_confirm_delay_on_connect, max_confirm_delay_on_data := dl_max_confirm_delay_on_data, dlsdu_size_from_requester := dl_dlsdu_size_from_requester, *1 dlsdu_size_from_responder := dl_dlsdu_size_from_responder, *2 dls_user_data := dl_dls_user_data *1 } *1: Not used with BNU AREPs whose role is Subscriber. *2: Not used with BNU AREPs whose role is Publisher.	ACTIVE
R6	ACTIVE	DL_Connection_Established.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state
R7	ACTIVE	DL_Connection_Established.ind && FindAREP (dl_dlcep_dl_id) = "True" => FAS-PDU_ind { dmpm_service_name := "DMPM_Connection_Established_ind", fas_pdu := "null" }	ACTIVE
R8	ACTIVE	DL_Disconnect.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE
R9	ACTIVE	DL_Disconnect.ind && FindAREP (dl_dlcep_dl_id) = "True" => FAS-PDU_ind { dmpm_service_name := "DMPM_Disconnect_ind", originator := dl_originator, reason := dl_reason, fas_pdu := dl_dls_user_data }	ACTIVE
R10	ACTIVE	DL_Unitdata.ind && FindAREP (dl_called_address) = "False" => (no actions taken)	ACTIVE
R11	ACTIVE	DL_Unitdata.ind && FindAREP (dl_called_address) = "True" && ExplicitQueue = "False" => FAS-PDU_ind { dmpm_service_name := "DMPM_Unitdata_ind", calling_address := dl_calling_address, dll_priority := dl_dll_priority, fas_pdu := dl_dls_user_data }	ACTIVE
R12	ACTIVE	DL_Unitdata.ind && FindAREP (dl_called_address) = "True" && ExplicitQueue = "True" => DL_Get.in { dl_buffer_or_queue_dl_id := QueueIdentifier } DL_Get.out && dl_status = "success" && dl_reported_service_identification_class = "DL(SAP)-ADDRESS" && dl_called_dl(sap)_address = LocalDisapAddress FAS-PDU_ind { dmpm_service_name := "DMPM_Unitdata_ind", calling_address := dl_calling_address, dll_priority := dl_dll_priority, fas_pdu := dl_dls_user_data } DL_Get.out && dl_status <> "success" ErrorToARPM { originator := "local_dls", reason := dl_status }	ACTIVE
R13	ACTIVE	DL_Unitdata.cnf && dmpm_request_id = dl_request_dls_user_id && dl_status <> "success" => ErrorToARPM { originator := "local_dls", reason := dl_status }	ACTIVE
R14	ACTIVE	DL_Unitdata.cnf && dmpm_request_id = dl_request_dls_user_id && dl_status = "success" => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state
R15	ACTIVE	DL_Data.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE
R16	ACTIVE	DL_Data.ind && FindAREP (dl_dlcep_dl_id) = "True" && ExplicitQueue = "False" => FAS-PDU_ind { dmpm_service_name := "DMPM_Data_ind", fas_pdu := dl_dls_user_data }	ACTIVE
R17	ACTIVE	DL_Data.ind && FindAREP (dl_dlcep_address) = "True" && ExplicitQueue = "True" => DL_Get.in { dl_buffer_or_queue_id := ReceivingBufferOrQueueIdentifier } DL_Get.out && dl_status = "success" && dl_reported_service_identification_class = "DLCEP" FAS-PDU_ind { dmpm_service_name := "DMPM_Data_ind", fas_pdu := dl_dls_user_data } DL_Get.out && dl_status <> "success" ErrorToARPM { originator := "local_dls", reason := dl_status }	ACTIVE
R18	ACTIVE	DL_Data.cnf && dmpm_request_id <> dl_request_dls_user_id => (no actions taken)	ACTIVE
R19	ACTIVE	DL_Data.cnf && dmpm_request_id = dl_request_dls_user_id && dl_status = "success" => (no actions taken)	ACTIVE
R20	ACTIVE	DL_Data.cnf && dmpm_request_id = dl_request_dls_user_id && dl_status <> "success" => ErrorToARPM { originator := "local_dls", reason := dl_status, dlsdu := "null" }	ACTIVE
R21	ACTIVE	DL_Buffer_Received.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)	ACTIVE

#	Current state	Event or condition => action	Next state
R22	ACTIVE	<pre>DL_Buffer_Received.ind && FindAREP (dl_dlcep_dl_id) = "True" => dmpm_duplicate_dlsdu := dl_duplicate_dlsdu, DL_Get.in { dl_buffer_or_queue_id := ReceivingBufferOrQueueIdentifier } DL_Get.out && dl_status = "success" && dl_reported_service_identification_class = "NONE" FAS-PDU_ind { dmpm_service_name := "DMPM_Buffer_Received_ind", duplicate_dlsdu := dmpm_duplicate_dlsdu, fas_pdu := dl_dls_user_data, local_dle_timeliness := dl_local_dle_timeliness, remote_dle_timeliness := dl_sender_and_remote_dle_timeliness } DL_Get.out && dl_status <> "success" ErrorToARPM { originator := "local_dls", reason := dl_status }</pre>	ACTIVE
R23	ACTIVE	<pre>DL_Buffer_Sent.ind && FindAREP (dl_dlcep_dl_id) = "False" => (no actions taken)</pre>	ACTIVE
R24	ACTIVE	<pre>DL_Buffer_Sent.ind && FindAREP (dl_dlcep_dl_id) = "True" => FAS-PDU_ind { dmpm_service_name := "DMPM_Buffer_Sent_ind" }</pre>	ACTIVE

10.3 Primitives exchanged between data-link layer and DMPM

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

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

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

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

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

10.4 Functions used by DMPM

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

Table 34 – Function PickArep

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

Table 35 – Function FindAREP

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

Table 36 – Function LocateQubArep

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

Table 37 – Function SetIdentifier()

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

Bibliography

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

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

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

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards-based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. Details and advice can be obtained from the Copyright & Licensing Department.

Useful Contacts:

Customer Services

Tel: +44 845 086 9001

Email (orders): orders@bsigroup.com

Email (enquiries): cservices@bsigroup.com

Subscriptions

Tel: +44 845 086 9001

Email: subscriptions@bsigroup.com

Knowledge Centre

Tel: +44 20 8996 7004

Email: knowledgecentre@bsigroup.com

Copyright & Licensing

Tel: +44 20 8996 7070

Email: copyright@bsigroup.com



...making excellence a habit.™