

DIN EN 61158-6-7

ICS 35.100.70; 35.160

Ersatzvermerk
siehe unten**Industrielle Kommunikationsnetze –
Feldbusse –****Teil 6-7: Protokollspezifikation des Application Layer
(Anwendungsschicht) – Typ 7-Elemente (IEC 61158-6-7:2007);
Englische Fassung EN 61158-6-7:2008, nur auf CD-ROM**Industrial communication networks –
Fieldbus specifications –Part 6-7: Application layer protocol specification – Type 7 elements (IEC 61158-6-7:2007);
English version EN 61158-6-7:2008, only on CD-ROMRéseaux de communication industriels –
Spécifications des bus de terrain –Partie 6-7: Spécification des services des couches d'application – Éléments de type 7
(CEI 61158-6-7:2007);
Version anglaise EN 61158-6-7:2008, seulement en CD-ROM**Ersatzvermerk**

Mit DIN EN 61158-6-2:2008-09, DIN EN 61158-6-3:2008-09, DIN EN 61158-6-4:2008-09,
DIN EN 61158-6-5:2008-09, DIN EN 61158-6-8:2008-09, DIN EN 61158-6-9:2008-09,
DIN EN 61158-6-10:2008-09, DIN EN 61158-6-11:2008-09, DIN EN 61158-6-12:2008-09,
DIN EN 61158-6-13:2008-09, DIN EN 61158-6-14:2008-09, DIN EN 61158-6-15:2008-09,
DIN EN 61158-6-16:2008-09, DIN EN 61158-6-17:2008-09, DIN EN 61158-6-18:2008-09,
DIN EN 61158-6-19:2008-09 und DIN EN 61158-6-20:2008-09 Ersatz für DIN EN 61158-6:2005-05
Siehe jedoch Beginn der Gültigkeit

Gesamtumfang 157 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Beginn der Gültigkeit

Die von CENELEC am 2008-02-01 angenommene EN 61158-6-7 gilt als DIN-Norm ab 2008-09-01.

Daneben darf DIN EN 61158-6:2005-05 noch bis 2011-02-01 angewendet werden.

Nationales Vorwort

Mit DIN EN 61158-6-2:2008-09, DIN EN 61158-6-3:2008-09, DIN EN 61158-6-4:2008-09, DIN EN 61158-6-5:2008-09, DIN EN 61158-6-8:2008-09, DIN EN 61158-6-9:2008-09, DIN EN 61158-6-10:2008-09, DIN EN 61158-6-11:2008-09, DIN EN 61158-6-12:2008-09, DIN EN 61158-6-13:2008-09, DIN EN 61158-6-14:2008-09, DIN EN 61158-6-15:2008-09, DIN EN 61158-6-16:2008-09, DIN EN 61158-6-17:2008-09, DIN EN 61158-6-18:2008-09, DIN EN 61158-6-19:2008-09 und DIN EN 61158-6-20:2008-09 Ersatz für DIN EN 61158-6:2005-05.

Vorausgegangener Norm-Entwurf: E DIN EN 61158-600:2006-11.

Für diese Norm ist das nationale Arbeitsgremium K 956 „Feldbus“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Die enthaltene IEC-Publikation wurde vom SC 65C „Industrial networks“ erarbeitet.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (maintenance result date) unverändert bleiben soll, das auf der IEC-Website unter „<http://webstore.iec.ch>“ zu dieser Publikation angegeben ist. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Einführung

Ein Feldbus ist ein digitaler serieller Datenbus für die Kommunikation zwischen Geräten der industriellen Automatisierungstechnik wie beispielsweise Messeinrichtungen, Stellgliedern, Reglern und speicherprogrammierbaren Steuerungen. Die Normen der Reihe DIN EN 61158 legen eine Anzahl von Feldbus-Protokolltypen fest.

Die vorliegende Norm legt das Protokoll des Application Layer für einen Feldbus des Typs 7 (WorldFIP) fest.

Änderungen

Gegenüber DIN EN 61158-6:2005-05 wurden folgende Änderungen vorgenommen:

- a) Streichung des Typ 6-Feldbusses wegen zu geringer Marktbedeutung;
- b) Hinzufügung von weiteren Feldbustypen;
- c) Aufteilung in mehrere Teile, jeweils als 6-2, 6-3, ..., 6-20 benummert.

Frühere Ausgaben

DIN V 19245-1: 1988-01
DIN 19245-1: 1991-04
DIN 19245-2: 1991-04
DIN EN 50170-1: 1997-07
DIN EN 50170-2: 1997-07
DIN EN 50170-3: 1997-07
DIN EN 50254: 1999-07
DIN EN 50170/A1: 2002-11
DIN EN 50170/A2: 2000-08
DIN EN 50170/A3: 2002-11
DIN EN 61158-6: 2005-05

– Leerseite –

Deutsche Fassung

Industrielle Kommunikationsnetze –
Feldbusse –
Teil 6-7: Protokollspezifikation des Application Layer (Anwendungsschicht) –
Typ 7-Elemente
(IEC 61158-6-7:2007)

Industrial communication networks –
Fieldbus specifications –
Part 6-7: Application layer protocol specification –
Type 7 elements
(IEC 61158-6-7:2007)

Réseaux de communication industriels –
Spécifications des bus de terrain –
Partie 6-7: Spécification des services des
couches d'application – Eléments de type 7
(CEI 61158-6-7:2007)

Diese Europäische Norm wurde von CENELEC am 2008-02-01 angenommen. Die CENELEC-Mitglieder sind gehalten, die CEN/CENELEC-Geschäftsordnung zu erfüllen, in der die Bedingungen festgelegt sind, unter denen dieser Europäischen Norm ohne jede Änderung der Status einer nationalen Norm zu geben ist.

Auf dem letzten Stand befindliche Listen dieser nationalen Normen mit ihren bibliographischen Angaben sind beim Zentralsekretariat oder bei jedem CENELEC-Mitglied auf Anfrage erhältlich.

Diese Europäische Norm besteht in drei offiziellen Fassungen (Deutsch, Englisch, Französisch). Eine Fassung in einer anderen Sprache, die von einem CENELEC-Mitglied in eigener Verantwortung durch Übersetzung in seine Landessprache gemacht und dem Zentralsekretariat mitgeteilt worden ist, hat den gleichen Status wie die offiziellen Fassungen.

CENELEC-Mitglieder sind die nationalen elektrotechnischen Komitees von Belgien, Bulgarien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Griechenland, Irland, Island, Italien, Lettland, Litauen, Luxemburg, Malta, den Niederlanden, Norwegen, Österreich, Polen, Portugal, Rumänien, Schweden, der Schweiz, der Slowakei, Slowenien, Spanien, der Tschechischen Republik, Ungarn, dem Vereinigten Königreich und Zypern.

CENELEC

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

Zentralsekretariat: rue de Stassart 35, B-1050 Brüssel

© 2008 CENELEC – Alle Rechte der Verwertung, gleich in welcher Form und in welchem Verfahren, sind weltweit den Mitgliedern von CENELEC vorbehalten.

Ref. Nr. EN 61158-6-7:2008 D

Vorwort

Der Text des Schriftstücks 65C/476/FDIS, zukünftige 1. Ausgabe von IEC 61158-6-7, ausgearbeitet von dem SC 65C „Industrial networks“ des IEC/TC 65 „Industrial-process measurement, control and automation“, wurde der IEC-CENELEC Parallelen Abstimmung unterworfen und von CENELEC am 2008-02-01 als EN 61158-6-7 angenommen.

Dieser und die weiteren Teile der Normenreihe EN 61158-6 ersetzen EN 61158-6:2004.

Gegenüber EN 61158-6:2004 wurden die folgenden Änderungen durchgeführt:

- Streichung des Typ 6-Feldbusses wegen zu geringer Marktbedeutung;
- Hinzufügung von weiteren Feldbustypen;
- Aufteilung in mehrere Teile, jeweils als 6-2, 6-3, ..., 6-20 benummert.

Nachstehende Daten wurden festgelegt:

- spätestes Datum, zu dem die EN auf nationaler Ebene durch Veröffentlichung einer identischen nationalen Norm oder durch Anerkennung übernommen werden muss (dop): 2008-11-01
- spätestes Datum, zu dem nationale Normen, die der EN entgegenstehen, zurückgezogen werden müssen (dow): 2011-02-01

ANMERKUNG Die Anwendung einiger der aufgeführten Protokolltypen wird durch die Inhaber der Rechte an geistigem Eigentum hieran eingeschränkt. Die Zusage zur begrenzten Freigabe ihrer Rechte an geistigem Eigentum durch die Inhaber erlaubt in jedem Fall die Anwendung eines bestimmten Data Link Layer-Protokolls zusammen mit einem Physical Layer- und einem Application Layer-Protokoll in Typzusammenstellungen, die in der Normenreihe EN 61784 angegeben sind. Die Anwendung der verschiedenen Protokolltypen in anderen Zusammenstellungen erfordert möglicherweise die Erlaubnis des jeweiligen Inhabers der Rechte an dem entsprechenden geistigen Eigentum.

Der Anhang ZA wurde von CENELEC hinzugefügt.

Anerkennungsnotiz

Der Text der Internationalen Norm IEC 61158-6-7:2007 wurde von CENELEC ohne irgendeine Abänderung als Europäische Norm angenommen.

In der offiziellen Fassung ist unter „Literaturhinweise“ zu der aufgelistete Norm die nachstehende Anmerkung einzutragen:

IEC 61784-1 **ANMERKUNG** Harmonisiert als EN 61784-1:2008 (nicht modifiziert).

CONTENTS

INTRODUCTION.....	7
1 Scope	8
1.1 General.....	8
1.2 Specifications	8
1.3 Conformance.....	8
2 Normative references	9
3 Terms, definitions, symbols, abbreviations and conventions	9
3.1 Terms and definitions from other ISO/IEC standards	9
3.2 Terms and definitions from IEC 61158-5-7	10
3.3 Additional terms and definitions	11
3.4 Abbreviations and symbols	15
3.5 Conventions	15
3.6 Conventions used in state machines	15
4 Abstract syntax of data type.....	16
4.1 Data abstract syntax specification	16
4.2 FAL PDU abstract syntax.....	20
5 Transfer syntaxes.....	21
5.1 Compact encoding.....	21
5.2 Data type encoding.....	22
6 Structure of protocol machines.....	80
7 AP-context state machine	81
8 Sub-MMS FAL service protocol machine (FSPM).....	81
8.1 General.....	81
8.2 Projection of the SUB-MMS PDUs on the MCS services	81
8.3 Projection of the SUB-MMS abort service on the MCS services	81
8.4 Construction of a SUB-MMS-PDU from a service primitive	82
8.5 Extraction of a valid service primitive from a SUB-MMS-PDU	82
8.6 Negotiation of an abstract syntax and a transfer syntax commonly called presentation-context	82
8.7 Identification of the SUB-MMS core abstract syntax.....	84
8.8 Identification of the application context name	85
8.9 Identification of the ASE of the core abstract syntax and the transfer syntax	85
9 Association relationship protocol machine (ARPM)	85
10 DLL mapping protocol machine (DMPM).....	85
10.1 MPS ARPM and DMPM.....	85
10.2 MCS ARPM and DMPM	97
11 Protocol options	132
11.1 Conformances classes.....	132
Bibliography	152
Annex ZA (normative) Normative references to international publications with their corresponding European publications.....	153
Figure 1 – Example of an evaluation net.....	16
Figure 2 – Encoding of a CompactValue.....	21

Figure 3 – Organisation of the bits and octets within a PDU	23
Figure 4 – Encoding of a Bitstring	26
Figure 5 – Encoding of a Floating point value	27
Figure 6 – Encoding of a structure	28
Figure 7 – Encoding of a Boolean array.....	29
Figure 8 – Representation of a MCS PDU	35
Figure 9 – Relationships among Protocol Machines and Adjacent Layers.....	80
Figure 10 – A_Readloc service evaluation net	86
Figure 11 – A_Writeloc service evaluation net	87
Figure 12 – A_Update service evaluation net.....	88
Figure 13 – A_Readfar service evaluation net	89
Figure 14 – A_Writefar service evaluation net	91
Figure 15 – A_Sent service evaluation net	92
Figure 16 – A_Received service evaluation net	92
Figure 17 – Association establishment: Requester element state machine	99
Figure 18 – Association establishment: Responder element state machine	100
Figure 19 – Association termination: Requester element state machine	102
Figure 20 – Association termination: Responder element state machine.....	104
Figure 21 – Association revocation: Requester element state machine.....	105
Figure 22 – Association revocation: Acceptor element state machine	106
Figure 23 – Interactions between state machine in an associated mode data transfer.....	107
Figure 24 – Transfer service: Requester element state machine	111
Figure 25 – Transfer service: Acceptor element state machine.....	112
Figure 26 – Unacknowledged transfer: Requester element state machine	113
Figure 27 – Unacknowledged transfer: Acceptor element state machine	113
Figure 28 – Acknowledged transfer: Requester element state machine	115
Figure 29 – Acknowledged transfer: Acceptor element state machine	116
Figure 30 – Numbering mechanism state machine	117
Figure 31 – Retry machanism state machine	119
Figure 32 – Anticipation mechanism state machine.....	121
Figure 33 – Segmentation mechanism state machine	124
Figure 34 – Reassembly mechanism state machine.....	126
Figure 35 – Interaction of state machine in a non associated data transfer.....	128
Figure 36 – Unacknowledged transfer: Requester element state machine	129
Figure 37 – Unacknowledged transfer: Acceptor element state machine	129
Figure 38 – Acknowledged transfer: Requester element state machine	130
Figure 39 – Acknowledged transfer: Acceptor element state machine	131
Table 1 – Example of encoding of a SEQUENCE.....	18
Table 2 – Example of encoding of a SEQUENCE OF	18
Table 3 – Example of encoding of a CHOICE	19
Table 4 – Example of encoding of an object identifier	20
Table 5 – Example of encoding of a PDU	21

Table 6 – MPS PDU types.....	24
Table 7 – Fields of a CompactValuePDU	24
Table 8 – Fields of a VariableDescriptionPDU	30
Table 9 – Fields of an AccessDescriptionPDU	32
Table 10 – Fields of a TypeDescriptionPDU.....	33
Table 11 – Fields of a ListDescriptionPDU.....	33
Table 12 – Coding of the different MCS PDU types.....	35
Table 13 – Coding of the variable part of the PDU	35
Table 14 – Structure of association establishment request.....	36
Table 15 – Structure of an associated establishment response	39
Table 16 – Structure of an association termination request	42
Table 17 – Structure of an association termination response.....	42
Table 18 – Structure of an association revocation request	43
Table 19 – Structure of an associated transfer request	43
Table 20 – Structure of an associated transfer acknowledgement	44
Table 21 – Structure of a non-associated transfer request	44
Table 22 – Structure of a non-associated transfer acknowledgement	45
Table 23 – Definitions of object classes.....	47
Table 24 – Definition of Sub-MMS Services	49
Table 25 – Structure of the antiduplication list	120
Table 26 – Structure of the reassembly list.....	125
Table 27 – PV_R/W parameter values	133
Table 28 – PV_IND parameter values.....	133
Table 29 – PV_LIS parameter values	133
Table 30 – Constraints on PV_LIS parameter	134
Table 31 – PV_AT parameter values	134
Table 32 – PV_RE parameter values.....	134
Table 33 – PV_UT parameter values	134
Table 34 – Constraints on PV_RE parameter.....	134
Table 35 – PH_R_A parameter values.....	135
Table 36 – PH_R_S parameter values.....	135
Table 37 – PH_R_P parameter values.....	135
Table 38 – PH_P_A parameter values	136
Table 39 – PH_P_S parameter values	136
Table 40 – PH_P_P parameter values	136
Table 41 – PH_COH parameter values	136
Table 42 – PH_FIA parameter values	137
Table 43 – PH_SPF parameter values.....	137
Table 44 – PH_SPM parameter values	137
Table 45 – PH_ACC parameter values	137
Table 46 – PH_RES parameter values	138
Table 47 – PH_AK parameter values.....	138
Table 48 – PH_RA parameter values.....	138

Table 49 – PH_SR parameter values	138
Table 50 – PH_CF parameter values.....	139
Table 51 – Constraints on PH_RA parameter	139
Table 52 – Constraints on PH_SR parameter	139
Table 53 – PT_OCT parameter values	139
Table 54 – PT_BIN parameter values	139
Table 55 – PT_VIS parameter values	140
Table 56 – PT_BOO parameter values	140
Table 57 – PT_BCD parameter values	140
Table 58 – PT_BTM parameter values	140
Table 59 – PT_INT parameter values	141
Table 60 – PT_UNE parameter values	141
Table 61 – PT_FPT parameter values	141
Table 62 – PT_GTM parameter values	141
Table 63 – PT_TAB parameter values	142
Table 64 – PT_STR parameter values.....	142
Table 65 – Constraints on PT_TAB parameter.....	142
Table 66 – Constraints on PT_STR parameter.....	143
Table 67 – Conformance classes for environment management.....	145
Table 68 – Conformance classes for VMD management	147
Table 69 – Conformance classes for PI managment	148
Table 70 – Conformance classes for domain management.....	149
Table 71 – Conformance classes for variable/variable list management	150
Table 72 – Conformance classes for event management	151

INTRODUCTION

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

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-7: Application layer protocol specification – Type 7 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 7 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard specifies interactions between remote applications and defines the externally visible behavior provided by the Type 7 fieldbus application layer in terms of

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

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

- define the wire-representation of the service primitives defined in IEC 61158-5-7, and
- define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 7 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-7.

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 parts of the IEC 61158-6 series.

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems.

There is no conformance of equipment to the application layer service definition standard. Instead, conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

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

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

IEC 61158-3-7, *Industrial communication networks – Fieldbus specifications – Part 3-7: Data-link layer service definition – Type 7 elements*

IEC 61158-4-7, *Industrial communication networks – Fieldbus specifications – Part 4-7: Data-link layer protocol specification – Type 7 elements*

IEC 61158-5-7, *Industrial communication networks – Fieldbus specifications – Part 5-7: Application layer service definition – Type 7 elements*

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

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

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

ISO/IEC 9506-2, *Industrial automation systems – Manufacturing Message Specification – Part 2: Protocol specification*

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

3 Terms, definitions, symbols, abbreviations and conventions

For the purposes of this document, the following definitions 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

- 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

- 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 Terms and definitions from IEC 61158-5-7

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

3.3 Additional terms and definitions

3.3.1

allocate

take a resource from a common area and assign that resource for the exclusive use of a specific entity

3.3.2

application

function or data structure for which data is consumed or produced

3.3.3

application objects

multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

3.3.4

attribute

description of an externally visible characteristic or feature of an object

NOTE 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.

3.3.5

behaviour

indication of how an object responds to particular events

3.3.6

called

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

3.3.7

calling

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

3.3.8

class

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

NOTE A class is a generalisation of an 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.

3.3.9

class attributes

attribute that is shared by all objects within the same class

3.3.10

class code

unique identifier assigned to each object class

3.3.11

class specific service

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

NOTE A class specific object is unique to the object class which defines it

3.3.12

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

3.3.13

clock

device providing a measurement of the passage of time since a defined epoch

NOTE There are two types of clocks in IEC 61588, boundary clocks and ordinary clocks.

3.3.14

communication objects

components that manage and provide a run time exchange of messages across the network

EXAMPLES Connection Manager object, Unconnected Message Manager (UCMM) object, Message Router object.

3.3.15

connection

logical binding between application objects that may be within the same or different devices

NOTE Connections may be either point-to-point or multipoint.

3.3.16

connection ID (CID)

identifier assigned to a transmission that is associated with a particular connection between producers and consumers, providing a name for a specific piece of application information

3.3.17

connection point

buffer which is represented as a subinstance of an Assembly object

3.3.18

consume

act of receiving data from a producer

3.3.19

consumer

node or sink that is receiving data from a producer

3.3.20

consuming application

application that consumes data

3.3.21

cyclic

repetitive in a regular manner

3.3.22

device

physical hardware connected to the link

NOTE: A device may contain more than one node.

3.3.23

device profile

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

3.3.24

end node

producing or consuming node

3.3.25

end point

one of the communicating entities involved in a connection

3.3.26

error

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

3.3.27

frame

denigrated synonym for DLPDU

3.3.28

instance

the actual physical occurrence of an object within a class, identifying one of many objects within the same object class.

EXAMPLE California is an instance of the object class state.

NOTE The terms object, instance, and object instance are used to refer to a specific instance.

3.3.29

instance attribute

attribute that is unique to an object instance and not shared by the object class

3.3.30

instantiated

object that has been created in a device

3.3.31

interoperability

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

3.3.32

little endian

Describes a model of memory organisation which stores the least significant octet at the lowest address, or for transfer, which transfers the lowest order octet first

3.3.33

management information

network accessible information that supports the management of the Fieldbus environment

3.3.34

member

piece of an attribute that is structured as an element of an array

3.3.35

message router

object within a node that distributes messaging requests to appropriate application objects

3.3.36

multipoint connection

connection from one node to many

NOTE Multipoint connections allow messages from a single producer to be received by many consumer nodes.

3.3.37

network

a set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.3.38

object

abstract representation of a particular component within a device, usually a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behaviour

3.3.39

object specific service

service unique to the object class which defines it

3.3.40

originator

client responsible for establishing a connection path to the target

3.3.41

point-to-point connection

connection that exists between exactly two application objects

3.3.42

produce

act of sending data to be received by a consumer

3.3.43

producer

node that is responsible for sending data

3.3.44

receiving

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

3.3.45

resource

resource is a processing or information capability of a subsystem

3.3.46

sending

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

3.3.47

service

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

3.4 Abbreviations and symbols

ASCII	American Standard Code for Information Interchange
CID	connection ID
DLL	data-link layer
PDU	protocol data unit
OSI	open systems interconnection (see ISO/IEC 7498)
Rcv	receive
Rx	receive
SDU	service data unit
STD	state transition diagram, used to describe object behaviour
TPDU	transport protocol data unit
Tx	transmit
Xmit	transmit

3.5 Conventions

3.5.1 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. 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.

3.5.2 Conventions for class definitions

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

3.5.3 Abstract syntax conventions

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

3.6 Conventions used in state machines

3.6.1 General

This subclause recalls the main definitions associated with the three specification techniques used : 'Evaluation Networks', 'Abstract objects', 'Service Conventions'.

3.6.2 Evaluation networks

To be helpful for the reader some mechanisms are explained by a graphical representation as shown in Figure 1. These representations are based on evaluation network built on an oriented graph with three types of nodes:

- the states, represented by a disk,
- the requests represented by a rectangle,

- the transitions, represented by a horizontal line.

The evaluation network is built according to the following principle:

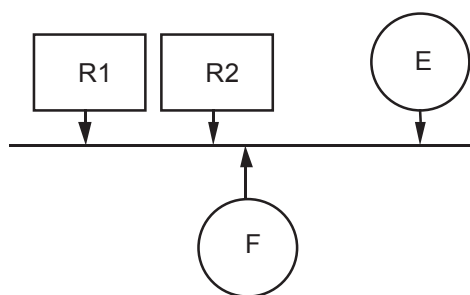


Figure 1 – Example of an evaluation net

When the described system is in state E, the arrival of request R1 or R2 results in a transition which switches it state F.

On the other hand, crossing a transition takes place, by convention, in zero time.

3.6.3 Simple actions

Simple actions can be associated with each transition, corresponding either to the transmission of requests or to the execution of local actions.

They are materially represented by a triangle, labelled with the name of the request or of the description of the executed action.

3.6.4 Conditions

The crossing of a transition can concern a condition. In this case, the oriented graph associated with the representation, shows a condition associated with the arc which precedes the transition.

A corollary of this representation allows defining choices for crossing a transition from the same initial state.

In the same way as for a simple transition, the transmission of an action can be associated with the crossing of a conditional transition.

4 Abstract syntax of data type

4.1 Data abstract syntax specification

This present standard uses the following terms defined in the abstract syntax number 1 (ASN1). Different or added definitions are the following :

4.1.1 Boolean

A Boolean value is encoded on an octet all of whose bits are null for FALSE; any other combination means TRUE.

Example of notation of the type:

Response::= BOOLEAN

Example of value notation and encoding:

Response value::= TRUE

value is encoded, e.g., FFh or 6Dh.

4.1.2 Integer

The INTEGER value is encoded as a sequence of M octets describing a binary number:

- either as a complement of 2 if the values can be positive and negative: the most and least significant bits are respectively bit 8 of the first octet and bit 1 of the last octet. Bit 8 of the first octet represents the integer sign. If the size is the number of bits of the integer (i.e., $\text{size} = M \cdot 8$), the value of the integer is between $-2^{(\text{size}-1)}$ and $+2^{(\text{size}-1)}-1$;

Example of notation of the type:

Signed8::= INTEGER (-128..127)

Example of value notation and encoding:

integer Signed8::= -128

integer is encoded: FFh.

- or as an absolute value if the values are all positive or all negative: the most and least significant bits are respectively bit 8 of the first octet and bit 1 of the last octet. If the size is the number of bits of the integer (i.e., $\text{size} = M \cdot 8$), the value of the integer is between $-2^{(\text{size})}$ and $+2^{(\text{size})}-1$.

Example of type notation:

Unsigned8::= INTEGER (0..255)

Example of value notation and encoding:

integer Unsigned8::= 255

integer is encoded: FFh.

4.1.3 Bit string

The value of BIT STRING is encoded on a sequence of M octets; the first bit of the binary string is aligned with bit 8 of the least significant octet of the sequence of octets and its number is zero. The number of unused bits shall be between bit 1 and 7 of the most significant octet of the sequence of octets. The true value of a bit of the bit string takes the binary value 1; unused bits can take the value 0 or 1.

Example of notation of the type:

Bitstring::= BIT STRING SIZE(32)

Example of value notation and encoding:

binaryvalue Bitstring::= '5F291CD0'H

binary value is encoded: 5Fh 29h 1Ch D0h

4.1.4 Octet string

The value of OCTET STRING is encoded on a sequence of M octets; the first octet of the OCTET STRING is aligned with the least significant octet of the sequence of octets.

Example of type notation:

OctetString::= OCTET STRING SIZE(4)

Example value notation and encoding:

octetvalue OctetString::= '5F291CD0'H

octetvalue is encoded: 5Fh 29h 1Ch D0h.

4.1.5 Sequence (sequence or sequence of)

SEQUENCE or SEQUENCE OF encoding is the juxtaposition of the element encoding which comprise it.

Example of type notation:

INFO1::= SEQUENCE {name String, ok BOOLEAN}

String::= OCTET STRING SIZE(5) --string of 5 octets

Example of value notation and encoding:

value INFO1::= {name "SMITH", ok TRUE}

value is encoded as shown in Table 1.

Table 1 – Example of encoding of a SEQUENCE

0006h	53h	4Dh	49h	54h	48h	FFh
SEQUENCE length	"S"	"M"	"I"	"T"	"H"	TRUE

The top line represents the encoding proper and the bottom line describes what is encoded:

Example of type:

INFO2::= SEQUENCE OF date

date::= OCTET STRING SIZE (8) --YYYYMMDD

Example of value notation and encoding:

value INFO2::= { { date "19571111"}, {date "19590717"} }

value is encoded as shown in Table 2.

Table 2 – Example of encoding of a SEQUENCE OF

00h	10h
number	of octets

31h	39h	35h	37h	31h	31h
"1"	"9"	"5"	"7"	"1"	"1"

31h	31h	31h	39h	35h	39h
"1"	"1"	"1"	"9"	"5"	"9"

30h	37h	31h	37h
"0"	"7"	"1"	"7"

In each row pair, the top line represents the encoding proper and the bottom line describes what is encoded. Each table is read from left to right.

4.1.6 Choice

The encoding of CHOICE is the encoding of the possibility selected in this CHOICE.

Example of type notation:

INFO::= CHOICE {name [0] String, age [1] Unsigned8}

String::= OCTET STRING SIZE(5) --string of 5 octets

Unsigned8::= INTEGER (0..127) -- unsigned 8 bit integer

Example value notation and encoding:

value INFO::= {name "SMITH"}

value is encoded as shown in Table 3.

Table 3 – Example of encoding of a CHOICE

80h	53h	4Dh	49h	54h	48h
identification	"S"	"M"	"I"	"T"	"H"

The top line represents the encoding proper and the bottom line describes what is encoded:

4.1.7 Null

No content encoding is associated with the null element.

Identification encoding is possible if NULL is a possibility of a choice (Cf. section 10.1.2.5.2).

Example of notation of the type:

```
Type-Room::= SEQUENCE {
    number Unsigned8,
    person CHOICE {
        name [0] String,
        default [1] NULL --room not in use
    }
}
```

Unsigned8::= INTEGER (0..127)

String::= OCTET STRING SIZE(12)

Example of value notation and encoding:

value-Room Type-Room::= {number 30h, person { default NULL}}

value-Room is encoded 0002h 30h 81h

Length encoding is compulsory if NULL is optional (in this case, the length is necessarily equal to zero).

4.1.8 Object identifier

Encoding is comprised of an ordered list of encoding of concatenated sub-identifiers.

Every sub-identifier is represented by a sequence of one or more octets. Every octet is encoded as follows.

- Bit 8 of each octet indicates that it is the last in the sequence.
- Bit 8 of the last octet is on one.
- Bit 8 of each previous octet is on zero.
- Bits 7 to 1 of the octets in the sequence collectively encode the sub-identifier, knowing that:

Bits 5, 6 and 7 are zero.

- Bits 1 to 4 are used to encode each digit of each sub-identifier. Encoding occurs in the form of an unsigned binary number.

Example: {iso standard 9506 part(2) mms-general-module-version1(2)} is encoded as shown in Table 4.

Table 4 – Example of encoding of an object identifier

81h	80h	09h	05h	00h
"iso"	"standard"	9	5	0

86h	82h	82h
6	part(2)	mms...(2)

In each table, the top line represents the encoding proper and the bottom line describes what is encoded. Each table is read from left to right.

4.1.9 Default

The keyword DEFAULT is ignored.

4.2 FAL PDU abstract syntax

This section illustrates the FER encoding rules specified in the present standard, by showing the representation in octets of an element defined in FIELDBUS ASN.1.

```

PDU ::= CHOICE {
    req [0] IMPLICIT PDUreq,
    rep [1] IMPLICIT PDUrep, ....
}

PDUrep ::= SEQUENCE {
    invokeID Unsigned32,
    Response }

Response ::= CHOICE {
    status
    getprog
    [0] IMPLICIT StatusResponse, ...
    [45] IMPLICIT GetProgramInvocationAttributesResponse, ...}

GetProgramInvocationAttributesResponse ::= SEQUENCE {
    pi_state
    listOfDomainId
    mmsdeletable
    reusable
    monitor
    executionargument
    [0] IMPLICIT PiState,
    [1] IMPLICIT ListOfObjectID,
    [2] IMPLICIT BOOLEAN,
    [3] IMPLICIT BOOLEAN,
    [4] IMPLICIT BOOLEAN,
    [5] IMPLICIT OCTET STRING OPTIONAL
}

PiState ::= Unsigned8

Unsigned8 ::= INTEGER (0..127) --8 bit unsigned integer

ListOfObjectID ::= SEQUENCE OF Identifier

Identifier ::= Unsigned16

Unsigned16 ::= INTEGER (0..32767) --16 bit unsigned integer

Unsigned32 ::= INTEGER(0..2147483647) --32 bit unsigned integer

The registrated value is:
{ rep {invokeID 1,
    getprog { pi-state 8,
    listOfDomainId { {1024}, {1025}, {1026}, {1027} },
    mmsdeletable TRUE,
    reusable FALSE,
    monitor TRUE,
    executionargument "ARGUMENT"}
}

```

Its representation is as follows: octet encoding is described using the two-line tables below in which the top line represents the encoding proper and the bottom line describes what is encoded. The tables are read from left to right. See Table 5 for details.

Table 5 – Example of encoding of a PDU

81h	00h	00h	00h	01h	ADh		
rep	invokeID	(32 b ...		...)	getprog		

08h	00h	08h	04h	00h	04h	01h	
pi-state	number	of octets	1024	(16 b)	1025	(16b)	

04h	02h	04h	03h	FFh	00h	FFh	00h
1026	(16b)	1027	(16b)	TRUE	FALSE	TRUE	number

08h	41h	52h	47h	55h	4Dh	45h	4Eh
of octets	"A"	"R"	"G"	"U"	"M"	"E"	"N"

54h							
"T"							

5 Transfer syntaxes

5.1 Compact encoding

The encoding rules applied to the PDU type values represented by using the ASN1 (ISO 8824) abstract notation syntax, yield the transfer syntax.

In the MPS environment, the values associated with the PDU types, are globally encoded according to the encoding rules of ASN1 (ISO/IEC 8825), with some restrictions.

The encoding rules which yield the MPS transfer syntax associate a set of octets with each of the different PDU types.

This octet sequence comprises three components:

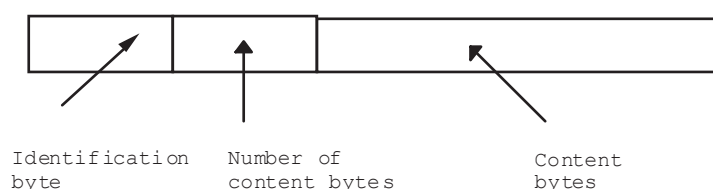
- identification
- size
- contents.

In the general case, the encoding rules permit contents which are recursively structured themselves in Identification, Size, Contents.

NOTE In the case of a CompactValue type PDU, the encoding has only one level.

The organization of an encoding level corresponds to that of the ISO/IEC 8825.

Encoding structure is shown in Figure 2.

**Figure 2 – Encoding of a CompactValue**

The following restrictions on the encoding relative to the MPS standard apply:

The Identification is encoded on 1 octet.

The Contents is encoded on a maximum of 126 octets.

5.2 Data type encoding

5.2.1 MPS ASE FAL PDU data types

5.2.1.1 Summary

The specification of the protocol provides a set of communication rules expressed in terms of data interchanges and procedures that shall be supported by the application entities.

AL TYPE 7 conformance is based on the compliance to these communication rules which are guidelines for conformance testing in order to achieve interoperability.

This protocol specification consists of:

- the description of the various protocol data units (PDU) interchanged between the communicating entities. This description is affected by use of an abstract syntax.
- encoding rules which transform the abstract syntax into transfer syntax.
- the description of the protocol procedures in terms of interactions between the primitives of each of the application services and the associated data link service primitives.

NOTE In the MPS environment, there is no direct interaction between a producer application entity and a consumer entity during the interchanges associated with a service primitive.

The specification of A_PDUs in the MPS environment is performed with a notation defined independently from the encoding. This notation constitutes the abstract syntax.

The abstract syntax is subsequently transformed by a set of encoding rules which are applied to the various types of A_PDU. These encoding rules yield the transfer syntax.

The conjunction of the abstract syntax and the encoding rules specifies the content of the octets interchanged between application entities by using the data link services.

NOTE The actual representation at the interface between the user and an application entity is out of the scope of this standard and is implementation dependent.

5.2.1.2 Syntax notation

The specification of the different A_PDUs associated with the MPS services is performed with the abstract notation syntax ASN1 defined in the ISO/IEC 8824.

All the ASN1 definitions of this standard are part of the module: " AL TYPE 7 - -MPS-1 ". All the grammar elements which constitute the Part5-3-MPS-1 module correspond to the first version of MPS.

Declaration of the AL TYPE 7 - -MPS-1 module:
AL TYPE 7 - -MPS-1 DEFINITIONS:= BEGIN

NOTE In order to meet the requirements of the ASN1 notation, all the keywords of the language are written in capital letters and the final symbol names begin with a capital letter. Furthermore, all the grammar items existing in this standard which take part in the definition of the AL TYPE 7 - -MPP-1 module are outlined with a double line before and after them.

General quote: The present standard specifies the value of each octet of an encoding by the sentences 'the most significant bit' and 'the least significant bit'. The bits of an octet are numbered from 8 to 1: bit 8 is 'the most significant bit' and bit 1 'the least significant bit'. This encoding is summarized in Figure 3.

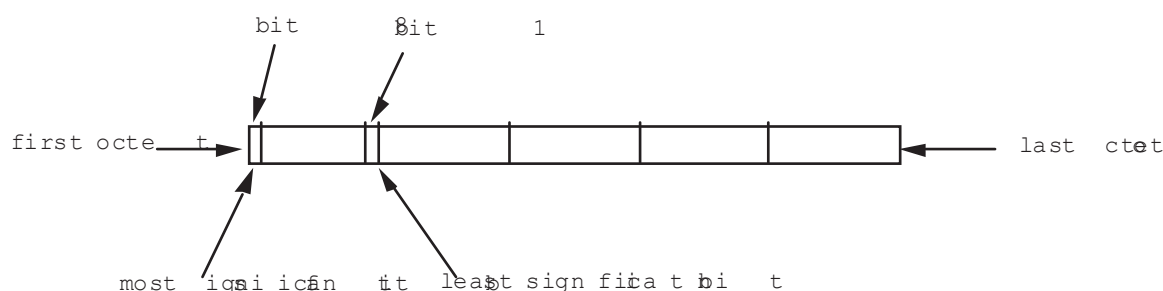


Figure 3 – Organisation of the bits and octets within a PDU

5.2.1.3 Simple predefined types

The PDU specification uses simple predefined types.

Some simple types are predefined by means of a statement that defines limits for these type values.

```
Unsigned4      ::= INTEGER          -- values limited between 0 and 15
Unsigned8      ::= INTEGER          -- values limited between 0 and 255
Unsigned16 ::= INTEGER              -- values limited between 0 and 65535
Symbol ::= Visible String           -- number of characters limited between 0 and 16
```

In the MPS environment the variable names, the access names, the type names and the list names are described as a combination of 16 character maximum length.

The characters belong to the set defined for the 'visible string' type of the ASN1 (ISO 8824) syntax notation.

In this set, the following characters are permitted:

Capital letters	(A..Z)
Letters	(a..z)
Digits	(0..9)
Underscore	(_)
Currency symbol (\$)	

The 'space' character is not permitted.

Strings may not begin with a number.

The 'Unsigned16', 'Unsigned8' and 'Unsigned4' types are used in this standard to describe the limits of the maximum values of the MPS types.

5.2.1.4 PDU Types

5.2.1.4.1 Top Level Definition

The PDUs specified in the MPS environment convey the information associated with *Variable* objects. The PDU type used depends directly on the value of the *Construction* and *A_Name* attributes of the instance of the *Type constructor* associated with the variable and is summarized in Table 6.

Table 6 – MPS PDU types

Construction		PDU type
SIMPLE		CompactValuePDU
ARRAY		CompactValuePDU
STRUCTURE		CompactValuePDU
PREDEFINED	A_Name = K_DVAR	VariableDescriptionPDU
	A_Name = K_DACCESS	AccessDescriptionPDU
	A_Name = K_DTYPE	TypeDescriptionPDU
	A_Name = K_DLIST	ListDescriptionPDU
EXPLICIT		ExplicitPDU

The encoding of values of these different kinds of PDU uses the transfer syntax associated with ASN1 part 2 (ISO 8825). Nevertheless, array compacting and compressing rules can be implemented to optimize the flow on the bus.

NOTE These compacting rules allow the representation of structured type variables by using an octet string.

Specification of PDU type used in MPS:

MPS-pdu:: = CHOICE

{		
CompactedVariableValue	[APPLICATION 0]	IMPLICIT CompactValuePDU,
ExplicitVariableValue	[APPLICATION 1]	IMPLICIT ExplicitPDU,
VariableDescription	[APPLICATION 2]	IMPLICIT VariableDescriptionPDU,
AccessDescription	[APPLICATION 3]	IMPLICIT AccessDescriptionPDU
TypeDescription	[APPLICATION 4]	IMPLICIT TypeDescriptionPDU,
ListDescription	[APPLICATION 5]	IMPLICIT ListDescriptionPDU,
extensions	[APPLICATION 6]	ANY
}		

NOTE The [APPLICATION 7] to [APPLICATION 15] type PDUs are reserved by this standard for a future definition of other PDU types. The [APPLICATION 16] to [APPLICATION 30] type PDUs are reserved for the use of the Network Management standard. Moreover, the ANY type PDU is used to allow the future definition of additional PDU types in companion standards.

5.2.1.4.2 VariableCompactValue PDU

5.2.1.4.2.1 General

This PDU represents the variable value of SIMPLE, STRUCTURE or ARRAY construction type, and optionally the associated production status. These types are used for variables of NORMAL or SYNCHRONIZATION class.

The PDU of CompactValuePDU type is represented by using the OCTET STRING type:

Compact-value-pdu:: = OCTET STRING

-- The total length of the PDU shall not exceed 128 octets.

NOTE The fields of this PDU permits encoding of the following attributes in the MPS application layer objects. See Table 7.

Table 7 – Fields of a CompactValuePDU

Field	Class	Attribute
Compact_value_PDU	Variable	Public value Transmitted status value

The ASN1 OCTET STRING type will be then encoded under the ASN1 (ISO 8825 standard) rules as specified in the chapter concerning the transfer syntax.

NOTE The representation of a MPS variable under an ASN1 OCTET STRING type is only possible considering the following restrictions.

- Producers and consumers of the variable have a prior knowledge of the MPS type of the variable, and do not need to transmit the type description with the value of the variable. The knowledge is available by access to the description variable and description type variables.
- The type of the variable has a fixed length and has no conditional or optional fields.

The representation of the public value associated with a variable by using compacting rules is performed from the first content octet of the CompactValuePDU.

The production status are represented on the last octet when this option is selected.

The PDU's length octet value represents the octet count used to encode the variable value and the production status octet when it exists.

The representation of the production status on this octet is performed as follows.

- The refreshment status is encoded on bit 1. This bit is set to '1' when the refreshment status is 'TRUE', and '0' when it is 'FALSE'.
- The punctual refreshment status is encoded on bit 2. This bit is set to '1' when the punctual refreshment status is 'TRUE', and '0' when it is 'FALSE'.
- The bits 3 to 8 are reserved.

The encoding of a given variable is represented with a constant number of octets derived from the type specification of this variable.

5.2.1.4.2.2 BOOLEAN value encoding rule

The boolean encoding is performed on a single octet.

If the value is

FALSE: the octet is 00_{16}

TRUE: the octet may take any value different from zero chosen by the producer entity.

5.2.1.4.2.3 INTEGER value encoding rule

The encoding of an integer value is performed on one or several octets. The maximum number of octets is fixed by the maximum size authorized for the INTEGER type, that is, 32 octets.

The number of octets used for encoding an INTEGER type corresponds to $E[(e-1)/8] + 1$ where E is the integer part, and e the value of the *size* attribute associated with this type.

The content octets shall be a two-complement binary number equal to the integer value and be composed of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each following octet, up to and including the last octet of the contents.

NOTE The value of a twos-complement binary number is derived by numbering the bits of the content octets, starting from bit 1 of the last octet and ending at bit 8 of the first octet. Each bit is assigned a numerical value of 2^{n-1} , where n is the bit position within the numbering sequence. The value of the twos-complement binary number is obtained by adding the numerical values assigned to each of the bits set to '1' (except for bit 8 of the first octet) from which the signed value from bit 8 of the first octet. is derived. The sign is negative when bit 8 of the first octet is set to '1'.

5.2.1.4.2.4 BITSTRING value encoding rule

The encoding of a bitstring value is performed on one or several octets as shown in Figure 4.

The maximum number of octets is fixed by the maximum size authorized for a BITSTRING type, that is, 32 octets.

The number of octets used for encoding a BITSTRING type corresponds to $E[(e-1)/8] + 1$ where E is the integer part and e the size of this type.

The bits of the bit string, starting from the first bit and ending at the last bit, shall be placed in sequence and in this order:

- in bits 8 to 1 of the first octet for the first 8 bits of the string,
- in bits 8 to 1 of the second octet for the 8 following bits,
- then in bits 8 to 1 of each following octet,
- followed by as many bits of the final octet as necessary always starting with bit 1.

The unused bits of the last octet have a value chosen by the producer entity.

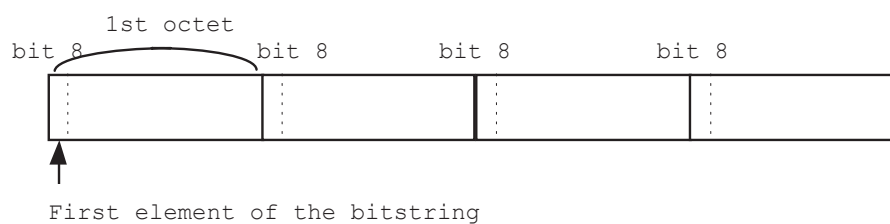


Figure 4 – Encoding of a Bitstring

5.2.1.4.2.5 UNSIGNED value encoding rule

The encoding of an unsigned integer value is performed on one or several octets. The maximum number of octets is fixed by the maximum size permitted for an UNSIGNED type, that is, 32 octets. The number of octets used for encoding a UNSIGNED type corresponds to $E[(e-1)/8] + 1$ where E is the integer part and e the value of the *size* attribute associated with this type.

The content octets shall be a binary number equivalent to the integer value, composed of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each following octet, up to and including the last octet of the contents.

NOTE The value of a binary number is obtained by numbering the bits of the content octets, starting from bit 1 of the last octet and ending at bit 8 of the first octet. Each bit is assigned a numerical value of 2^{n-1} , where n is the bit position within the numbering sequence.

5.2.1.4.2.6 OCTETSTRING value encoding rule

The encoding of an OCTETSTRING value contains zero, one or several content octets whose value is equal to that of the data value octets, in the same sequence as they are placed in the data value, the most significant bit of a data value being aligned with the most significant bit of a octet of the contents.

The number of octets used for encoding an OCTETSTRING type corresponds directly to the number of terms of this string within the limit of 126 octets.

5.2.1.4.2.7 VISIBLE STRING value encoding rule

The value of a visible string is composed of a string of characters from the set specified in ISO 8824. Each of the characters of this string is encoded in one octet.

The number of octets used for encoding a VISIBLE STRING type corresponds directly to the number of elements of the string within the limit of 126 octets.

5.2.1.4.2.8 GENERALIZED TIME value encoding rule

The encoding of the value is performed in the same manner as a 14 character VISIBLE STRING value.

The generalized time (specified in ISO 8824) is characterized by a specific number of characters encoded on octets.

The generalized time is a juxtaposition of characters in the following sequence:

- 4 digits for the year
- 2 digits for the month
- 2 digits for the day
- 2 digits for the hour
- 2 digits for the minutes
- 2 digits for the seconds.

The first octets are those including the encoding of the year.

The generalized time expresses directly a local time.

5.2.1.4.2.9 FLOATING POINT value encoding rule

The encoding of a floating point value is performed on 4 or 8 octets according to the size attribute corresponding to single or double precision.

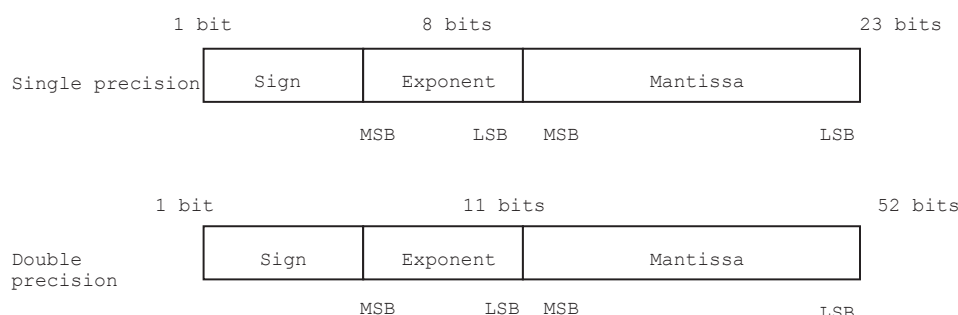


Figure 5 – Encoding of a Floating point value

The presentation formats defined in IEC 60559 are shown in Figure 5.

The content octets contain the sign, the exponent and the mantissa in the previous order starting from the first to the last octet.

In all cases the sign is encoded by using bit 8 of the first octet. It is followed by the the exponent starting from bit 7 of the first octet, and then the mantissa starts from bit 7 of the second octet for the simple precision and from bit 4 of the second octet for the double precision.

The encoding of the mantissa value and the exponent value is described in IEC 60559.

5.2.1.4.2.10 BINARY TIME value encoding rule

The encoding of a binary time value is performed on a number of octets defined by the size, and describes a number of elementary time units. The value of the elementary time unit depends on the selected size.

The content octets contain a binary value equivalent to the binary time value, and composed of bits 8 to 1 of the first octet, followed by bits 8 to 1 of each following octet, up to and including the last octets of the contents.

NOTE The binary value is obtained by numbering the bits of the content octets, starting from bit 1 of the last octet and ending at bit 8 of the first octet. Each bit set to '1' is assigned a numerical value of 2^{n-1} , where n is the bit position within the numbering.

5.2.1.4.2.11 BCD value encoding rule

The BCD encoding is performed on one octet. The binary value associated with the BCD value is encoded on bits 1 to 4 of the octet and obtained by starting from bit 1 up to bit 4. Each bit set to '1' is assigned a numerical value of 2^{n-1} , where n is the position of the bit within the octet.

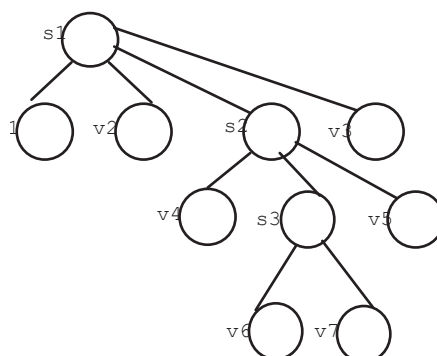
5.2.1.4.2.12 Structure encoding rule

The encoding of a structure type involves the encoding rules of each of the simple types and a concatenation rule of the types composing this structure.

The encoding of the various types composing a structure are juxtaposed in order of appearance within the structure.

The value associated with the first structure field is encoded in the first octets of the content. The value associated with the second field is encoded in the following octets, and so on, up to the last field of the structure.

Consider the following type tree:



Representation within the octets of the contents:



Figure 6 – Encoding of a structure

A structure can comprise several levels of abstraction; in this case, the encoding juxtaposition of the values associated with the various fields is performed by a first in depth traverse over the type tree associated with the structure as shown in Figure 6.

5.2.1.4.2.13 Array encoding

The array encoding involves the encoding rule of a simple type and a concatenation rule of the elements which compose the array.

The encoded values of the array elements are juxtaposed in the statement sequence of the array: these elements may be of simple, structure or array type.

The first array element is contained in the first octet of the content octets.

The array types may be optionally specified with optimisation rules of the associated encoding.

These optimisation rules do not preclude a partial access to an array element and preserve the integrity of the handled data.

5.2.1.4.2.14 Compacted encoding rule for a BOOLEAN array

The normal encoding of a BOOLEAN array type value uses one octet for each element of the array.

The compacted encoding of a BOOLEAN array type value is performed using one octet for a set of eight booleans.

Thus, the content octets used for encoding a compacted BOOLEAN array are organized as shown in Figure 7.

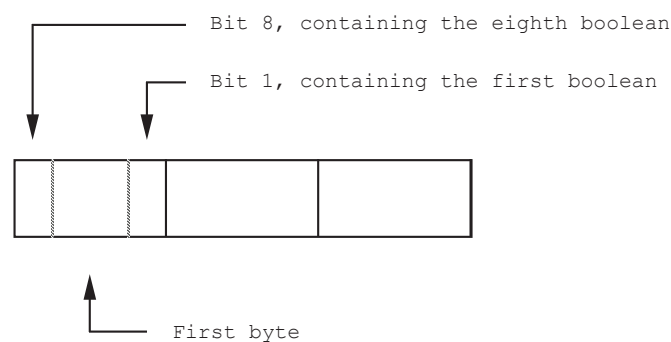


Figure 7 – Encoding of a Boolean array

In the compacted encoding of a boolean array, a boolean value is 'TRUE' when the corresponding bit is set to '1'.

Every compacted boolean array is encoded using a number of octets equal to $E[(e-1)/8] + 1$ where E is the integer part and e is the number of elements of this boolean array.

5.2.1.4.2.15 Compacted encoding rule for a BCD array

The normal encoding of a BCD array type value uses one octet for each element of the array.

The compact encoding of a BCD array type value is performed by using one octet for a pair of BCDs.

Thus, within the content octets used for encoding a compacted BCD array, bits 8 to 5 of the first octet contain the first BCD of the array, and bits 4 to 1 of the first octet contain the second BCD of the array, and so on up to the last BCD of the array.

The binary value associated with the value of the first BCD of a octet is obtained by starting from bit 5 up to bit 8: each bit is assigned a numerical value of 2^{n-5} , where n is the position of the bit within the octet.

The binary value associated with the value of the second BCD of a octet is obtained by starting from bit 1 up to bit 4: each bit is assigned a numerical value of 2^{n-1} , where n is the position of the bit within the octet.

5.2.1.4.3 ExplicitVariableValue PDU

The Explicit PDU type PDU allows the representation of the values of the variables for which the type construction is EXPLICIT.

This PDU supports variable values which are not of constant length.

```
ExplicitPDU ::= CHOICE
{
    array                [0] IMPLICIT SEQUENCE OF ExplicitPDU,
    structure            [1] IMPLICIT SEQUENCE OF ExplicitPDU,
    boolean              [2] IMPLICIT BOOLEAN,
    bitString            [3] IMPLICIT BITSTRING,
    integer              [4] IMPLICIT INTEGER,
    unsigned             [5] IMPLICIT INTEGER,
    floating             [6] IMPLICIT OCTET STRING,
    octetString          [7] IMPLICIT OCTET STRING,
    visibleString        [8] IMPLICIT OCTET STRING,
    generalizedTime      [9] IMPLICIT OCTET STRING,
    binaryTime           [10] IMPLICIT OCTET STRING,
    bcd                  [11] IMPLICIT INTEGER
}
```

The total length of the PDU shall not exceed 128 octets.

5.2.1.4.4 VariableDescription PDU

This PDU represents the value of the variables whose type is K_DVAR. This type is used for the description variables.

```
VariableDescriptionPDU ::= SEQUENCE
{
    FixedDescription      [0] IMPLICIT Description,
                        -- predefined type specified hereafter
    variableName          [1] IMPLICIT Symbol,
    resynchronisationReference [2] IMPLICIT ObjectReference OPTIONAL,
    productionPeriod      [3] IMPLICIT Unsigned16 OPTIONAL,
    synchronisationReference [4] IMPLICIT ObjectReference OPTIONAL,
    productionSlot        [5] IMPLICIT Unsigned16 OPTIONAL,
    punctualSynchroReference [6] IMPLICIT ObjectReference OPTIONAL,
    listReference          [7] IMPLICIT ObjectReference OPTIONAL,
    otherDetails           [8] IMPLICIT ANY OPTIONAL,
    vendorInformation      [9] IMPLICIT OCTET STRING OPTIONAL
}
```

The total length of the PDU shall not exceed 128 octets. See Table 8.

NOTE The fields of this PDU permits encoding of the following attributes in the MPS application layer objects.

Table 8 – Fields of a VariableDescriptionPDU

Field	Class	Attribute
FixedDescription	Variable	Identifier Network Period Class Consistency Variable Significant Status Reference Type Constructor
Variable Name	Variable	A_Name
ResynchronizationReference	Resynchronization in Production	Reference (Synchronization) Variable
ProductionPeriod	Refreshment	Production Variable
SynchronizationReference	Refreshment	Reference (Synchronization) Variable
ProductionSlot	Punctual Refreshment	Production Time Slot
PunctualSynchroReference	Punctual Refreshment	Reference (Synchronization) Variable
ListReference	Variable	Reference Variable List

The type description of this PDU uses the Description type which contains the non optional part of constant size of this PDU.

Description::= OCTET STRING -- Size fixed to 7 octets

Octet semantics:

octets 1 + 2: Described variable identifier

(Most significant bit of the identifier in bit 8 of the first octet and least significant bit in bit 1 of the second octet)

octets 3 + 4: Network period

(Most significant bit of the network period in bit 8 of the first octet, in multiples of 1 millisecond; a 0 value indicates an aperiodic variable)

octet 5: *Bit 8, and 7:* Variable class

00: NORMAL

01: SYNCHRONIZATION

10: DESCRIPTION

All the other combinations are reserved

Bit 6: Sub-class of a synchronization variable

1: Consistency variable

0: Synchronization variable but not consistency variable

Bits 5 and 4: Significant status

00: no significant status

01: significant refreshment

10: significant punctual refreshment

11: significant refreshment and punctual refreshment

Bits 3 to 1 reserved.

octets 6 + 7: Type reference

(encoded on 16 bits by using the description variable identifier of this type. The most significant bit is bit 8 of the octet 6 and the least significant one bit 1 of the octet 7)

The type description of this PDU uses the non terminal grammar element ObjectReference referring to variables, lists and types by means of the identifier of the associated description variable. These identifiers are encoded on two octets: the most significant bit of the identifier is encoded in bit 8 of the first octet and the least significant one in bit 1 of the second octet.

ObjectReference::= OCTET STRING -- Size fixed to 2 octets

The ANY types included in the definition of the PDU, as well as the reserved bits within this PDU, used to support, the description complements specified elsewhere.

Moreover, optional octet strings permit, insertion of vendor specific information into this PDU.

5.2.1.4.5 AccessDescription PDU

This PDU supports the variable value whose type is K_DACCESS. This type is used for the access description variables. See Table 9.

AccessDescriptionPDU ::= SEQUENCE

```
{
  accessName                [0] IMPLICIT Symbol,
  rootReference              [1] IMPLICIT ObjectReference,
  accessPath                 [2] IMPLICIT SEQUENCE OF CHOICE
  {
    structureFieldName       [0] IMPLICIT Symbol,
    arrayElementIndex        [1] IMPLICIT Unsigned8
  } OPTIONAL
  otherDetails               [3] IMPLICIT ANY OPTIONAL,
  vendorInformation          [4] IMPLICIT OCTET STRING OPTIONAL
}
```

NOTE The fields of this PDU permits encoding of the following attributes in the MPS application layer objects.

Table 9 – Fields of an AccessDescriptionPDU

Field	Class	Attribute
Access Name	Access	A_Name
Root Reference	Access	Reference Variable
Access Path	Type Constructor	Field Name

The ANY types include description complements specified elsewhere in the definition of the PDU.

Moreover, optional octet strings permit insertion of vendor specific information into this PDU.

5.2.1.4.6 TypeDescription PDU

This PDU supports the variable value whose type is K_DTYPE. This type is used for the type description variables.

TypeDescriptionPDU ::= SEQUENCE

```
{
  typeName                  [0] IMPLICIT Symbol,
  typeDescription           [1] CHOICE
  {
    array                   [0] IMPLICIT SEQUENCE
    {
      compressed            [0] IMPLICIT BOOLEAN DEFAULT FALSE,
      dimension              [1] IMPLICIT Unsigned8,
      elementReference       [2] IMPLICIT ObjectReference
    },
    structure                [1] IMPLICIT SEQUENCE OF SEQUENCE
    {
      fieldName              [0] IMPLICIT Symbol OPTIONAL,
      fieldTypeReference     [1] IMPLICIT ObjectReference
    },
    -- simple                Size                Type Class
    boolean                  [2] IMPLICIT NULL,          -- BOOLEAN
    binaryString              [3] IMPLICIT Unsigned8, -- BITSTRING
    integer                   [4] IMPLICIT Unsigned8, -- INTEGER
    unsigned                  [5] IMPLICIT Unsigned8, -- UNSIGNED
    floatingPoint             [6] IMPLICIT BOOLEAN, -- FLOATING POINT
    octetString               [7] IMPLICIT Unsigned8, -- OCTET STRING
    visibleString              [8] IMPLICIT Unsigned8, -- VISIBLE STRING
    universalTime              [9] IMPLICIT NULL,          -- UNIVERSAL TIME
    binaryTime                [10] IMPLICIT Unsigned4,    -- BINARY TIME
    bcd                       [11] IMPLICIT NULL,          -- BCD
    predefined                 [12] IMPLICIT NULL,
    explici                   [13] IMPLICIT NULL,
  },
  otherDetails               [2] IMPLICIT ANY OPTIONAL,
  vendorInformation           [3] IMPLICIT OCTET STRING OPTIONAL
}
```

-- The total length of the PDU shall not exceed 128 octets. See Table 10.

NOTE The fields of this PDU permits encoding of the following attributes in the MPS application layer objects.

Table 10 – Fields of a TypeDescriptionPDU

Field	Class	Attribute
TypeName	TypeConstructor	A_Name
Compressed	TypeConstructor	Compressed
Dimension	TypeConstructor	Dimension
ElementReference	TypeConstructor	TypeConstructor
FieldName	TypeConstructor	FieldName
FieldTypeReference	TypeConstructor	TypeConstructor
BinaryString	TypeConstructor	Size
Integer	TypeConstructor	Size
Unsigned	TypeConstructor	Size
FloatingPoint	TypeConstructor	Size
OctetString	TypeConstructor	Size
VisibleString	TypeConstructor	Size
BinaryTime	TypeConstructor	Size

The ANY types include description complements specified elsewhere in the definition of the PDU.

Moreover, optional octet strings permit insertion of information specific to the vendors into this PDU.

ListDescription PDU

This PDU supports the variable value whose type is K_DLIST. This type is used for the list description variables.

```

ListDescriptionPDU ::= SEQUENCE
{
  listName                [0] IMPLICIT Symbol,
  variableReferenceList    [1] IMPLICIT ListComponents,
  inconsistencyDetection    [2] IMPLICIT Unsigned8 OPTIONAL,
                           -- 0UNDEFINED
                           -- 1PERMANENT
                           -- 2PERMANENT
                           -- The other values are reserved
  synchronizationReference [3] IMPLICIT ObjectReference OPTIONAL,
  consistencyVariableList  [4] IMPLICIT ListConsistencyVariables OPTIONAL,
  recoveryPeriod           [5] IMPLICIT Unsigned16 OPTIONAL,
  otherDetails             [6] IMPLICIT ANY OPTIONAL,
  vendorInformation        [7] IMPLICIT OCTET STRING OPTIONAL
}

```

-- The total length of the PDU shall not exceed 128 octets. See Table 11.

NOTE The fields of this PDU permits encoding of the following attributes in the MPS application layer objects.

Table 11 – Fields of a ListDescriptionPDU

Field	Class	Attribute
ListName	VariableList	A_Name
VariableReferenceList	VariableList	List of Reference ConsumedVariable
InconsistencyDetection	VariableList	InconsistencyDetection
SynchronizationReference	VariableList	Reference (Synchronization)Variable
ConsistencyVariableList	VariableList	List of Reference (Consistency) Variable
RecoveryPeriod	VariableList	RecoveryPeriod

The type description of this PDU uses the non terminal grammar element 'listComponents' which defines the representation format of the references to the variables which compose the described list.

ListComponents ::= OCTET STRING -- The size of this string depends on that of the list.

Each variable name which comprise this list is encoded on two octets of this octet string, starting from the first two octets up to the last two octets within the size limits provided by this PDU. The list variable names are represented on two octets in a unique way by means of the identifier of the

description variable associated with each variable of this list. The most significant bit of the identifier is encoded in bit 8 of the first octet, and the least significant one in bit 1 of the second octet.

The type description of this PDU uses the non terminal grammar element 'listConsistencyVariables' which defines the representation format of the references to the consistency variables associated with the variable list in the case of the operation of the spatial consistency mechanism.

ListConsistencyVariables::= OCTET STRING -- size relative to the number of application entities involved in the spatial consistency.

Each consistency variable name, which takes part in the elaboration of the spatial consistency of this list, is encoded on two octets of this octet string, starting from the first two octets up to the last two octets within the size limits provided by this PDU. The consistency variable names are represented on two octets in an unique way by means of the identifier of the description variable associated with each consistency variable.

The most significant bit of the identifier is encoded in bit 8 of the first octet, and the least significant one in bit 1 of the second octet.

5.2.1.4.7 PDU Extensions

The ANY types include description complements specified elsewhere in the PDU definition.

For, all the grammar elements participating in the definition of the module being defined, it is necessary to declare the end of this module.

END -- End of the module AL TYPE 7 - -MPS-1

5.2.2 MCS AR ASE AL PDU types

5.2.2.1 Coding rules

All MCS PDUs contain a whole number of octets. The octets of a PDU are numbered from 1 to a maximum of 256 for the longest PDUs. These octets are transmitted via the bus in order from the first octet to the last. Each of the octets of a PDU has bits numbered 1 to 8, where bit 1 is the least significant and bit 8 the first transmitted on the line. Finally, when a number of consecutive octets represents a binary number, the least significant octet contains the most significant value.

In the following figures, the least significant number octets are always on the left when represented horizontally, and at the top when represented vertically. Also, the bits in a octet are always represented with bit 8 on the left and bit 1 on the right.

5.2.2.2 Structure of the PDUs

5.2.2.2.1 Top Level Definition

As a general rule, all MCS PDUs consist, in order, of:

- the header containing:
- the header size,
- the fixed part containing what is mandatory,
- the variable part containing what is optional,
- the data, which are optionally present.

Representation is shown in Figure 8

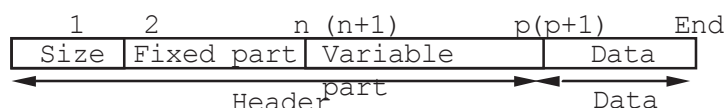


Figure 8 – Representation of a MCS PDU

a) Size:

The size is equal to sum of the numbers of octets of the fixed part and the variable part of the PDU header.

The value of the size is coded in the first octet of the header as a binary number.

The maximum size value is 255, which is coded as 1111 1111.

b) Fixed part

Whatever the PDU type, the fixed part contains its code followed by coding of a list of parameters which are mandatory for the code value.

The values of the PDU codes associated with the different types of PDUs are as shown in Table 12.

Table 12 – Coding of the different MCS PDU types

PDU values	Meaning
1000 0000	association establishment request
1100 0000	association establishment response
1001 0000	association termination request
1101 0000	association termination response
1011 0000	association revocation request
00X0 0XXX	associated transfer
0110 0000	associated transfer acknowledgement
00X1 0XXX	non-associated transfer
0111 0XXX	non-associated transfer acknowledgement

The bits of certain codes marked X have the values indicated in the descriptions of the associated PDU types.

NOTE All other codes are pending in this standard.

c) Variable part

This field is intended to receive optional parameters, which may be present or absent depending on the classes of conformity of the devices and the requirements of the user.

Each service parameter encoded in the variable part is structured as shown in Table 13.

Table 13 – Coding of the variable part of the PDU

Mapping	Meaning
octet 1	Parameter code
octet 2	Parameter length (length = m)
octet 2 + 1 (to)	Parameter value start (continued)
octet 2 + m	Parameter value end

Each of the parameters has a parameter code which is universal in the MCS environment, and which is therefore covered by this standard (these are defined for each PDU type implementing them).

The length of the parameter is coded in one octet. The value of the parameter is coded in as many octets as is specified in the length and the semantics of the coding is specific to each parameter (the semantics are defined at the level of each type of PDU implementing them).

d) Data

This field is transparent for MCS as it contains data specific to one of the specific application service elements projected onto MCS.

5.2.2.2.2 Association establishment request (AARQ)

AARQ structure is as shown in Table 14.

Table 14 – Structure of association establishment request

Mapping	Meaning
octet 1	Size
octet 2	PDU code
octet 3	Calling AEI access point
octet 4	ditto
octet 5	ditto
octet 6	Called AEI access point
octet 7	ditto
octet 8	ditto
octet 9	Class of conformity
octet 10	Variable part
to	ditto
octet p	ditto
octet p+1	User data
to	ditto
end octet	ditto

a) Size

The minimum size value is 8 (coded: 0000 1000) when the parameters relating to this PDU are not present, and the maximum size value is reached when all the parameters are present.

(For instance, the maximum size value is 51 octets.)

b) Fixed part

The following fields are present in the fixed part.

- The PDU code takes the value "1000 0000".
- The calling AEI access point, which corresponds to a link address (of an ADAEI type) is represented by bit I/G as the high order bit in the lowest number octet.

Representation:

```

octet_3:    I / G x x x    x x x x
octet_4:    x x x x x x x x
octet_5:    S / R x x x    x x x x

```

- The called AEI access point which corresponds to a liaison address (of ADAEI type) is represented with the high order bit in the lowest number octet.

Representation:

```

octet_6 :    I / G x x x    x x x x
octet_7 :    x x x x x x x x
octet_8 :    S / R x x x    x x x x

```

NOTE The default value used for this access point when the association establishment initiator does not wish to fill it in is the following value:

```

octet_6 :    1111    1111
octet_7 :    1111    1111
octet_8 :    1111    1110

```

It is a link address value which is found in the generalised group address space reserved under the standard.

- The class of conformity

Representation:

octet_9: b8 b7 b6 b5 b4 b3 b2 b1

Where:

b8 = 1, if PV_RE = 2
b7 = 1, if PH_AK = 2
b6 = 1, if PH_RA = 2
b5 = 1, if PH_SR = 2
b4 = 1, if PH_CF = 2

The possible combinations of these bits shall allow for the constraints placed on the classes of conformity

c) Variable part

The following parameters shall be present in the variable part.

- Transfer rate:

Parameter code: 0000 0000
Parameter length: 4 octets (0000 0100)
Parameter value:
 xxxx xxxx : Calling to Called
 xxxx xxxx
 yyyy yyyy : Called to Calling
 yyyy yyyy

The value of these two rates is represented by a binary number equal to the time interval expressed in ms between successive messages.

The value 0000 0000 0000 0001 corresponds to 1msg/ms and the value 1110 1010 0110 0000 corresponds to 1msg/min.

The value 0000 0000 0000 0000 conventionally represents a null guaranteed minimum flow in one direction of transfer.

This parameter can only be present if b4 (PH_CF) of the Class of conformity parameter is set to 1.

- Anticipation factor:

Parameter code: 0000 0001
Parameter length: 2 octets (0000 0010)
Parameter value:
 xxxx xxxx : Calling to Called
 yyyy yyyy : Called to Calling

The anticipation factor is represented by a binary number making it possible to discriminate between 255 levels. The anticipation level of "i" means that it is possible to have a maximum of "i" ATAK_SM_RQ current in the association.

The use of value 0 is illegal.

This parameter can only be present if the b7(PH_AK) of the Class of conformity parameter is 1 .

- Application context:

Parameter code: 0000 0010

Parameter length: 3 octets (0000 0011)

Parameter value:

xxxx	xxxx	:	Specific ASE type
yyyy	yyyy	:	Abstract syntax type
zzzz	zzzz	:	Transfer syntax type

The value of these different types is specific to each of the application ASEs. The high order bit of each of these octets indicates whether it is a type defined in a standard (b8=1) or specific to an application (b8=0).

- Calling entity identification:

Parameter code: 0000 0011

Parameter length: 6 octets (0000 0110)

Parameter value:

xxxx	xxxx	:	AEI identifier
yyyy	yyyy	:	API identifier
zzzz	zzzz	:	AE qualifier
uuuu	uuuu	:	AP title
uuuu	uuuu		
uuuu	uuuu		

The first octet of the value designates the AEI identifier by means of a binary number making it possible to discriminate between 255 AEIs for an API in the context of a given AE. Value 0000 0000 is the default value used to not fill in this parameter.

The second octet designates the API identifier by means of a binary number making it possible to discriminate between 255 APIs for a given AP. The value 0000 0000 is the default value used to not fill in this parameter.

The third octet designates the AE qualifier by means of a binary number making it possible to discriminate between 255 AEs for a given AP. The value 0000 0000 is the default value used to not fill in this parameter.

The last free octets designate the AP title by means of a binary number. These three octets are set to 0000 0000 if this parameter is not adopted.

- Called entity identification:

Parameter code: 0000 0100

Parameter length: 6 octets (0000 0110)

Parameter value:

xxxx	xxxx	:	AEI identifier
yyyy	yyyy	:	API identifier
zzzz	zzzz	:	AE qualifier
uuuu	uuuu	:	AP title
uuuu	uuuu		
uuuu	uuuu		

The first octet of the value designates the AEI identifier by means of a binary number making it possible to discriminate between 255 AEIs for one API in the context of a given AE. The value 0000 0000 is the default value used to not fill in this parameter.

The second octet designates the API identifier by means of a binary number making it possible to discriminate between 255 APIs for a given AP. The value 0000 0000 is the default value used to not fill in this parameter.

The third octet designates the AE qualifier by means of a binary number making it possible to discriminate between 255 AEs for a given AP. The value 0000 0000 is the default value used to not fill in this parameter.

The last three octets designate the AP title by means of a binary number. These three octets are set to 0000 0000 if this parameter is not adopted.

- SDU size:

Parameter code: 0000 0101

Parameter length: 2 octets (0000 0010)

Parameter value:

xxxx xxxx : Calling to Called

yyyy yyyy : Called to Calling

SDU size is represented by a binary number providing for a maximum SDU size of 256 DLPDUs, i.e. 64 koctets for a maximum DLPDU size of 256 octets.

Value 0 of this parameter is illegal.

This parameter may only be present if b5(PH_SR) of the Class of conformity parameter is set to 1.

- PDU size:

Parameter code: 0000 0110

Length: 1 octet (0000 0001)

Parameter value:

xxxx xxxx : SDU size

PDU size is represented by a binary number allowing a maximum PDU size of 256 octets.

This parameter can be present whatever the value of the class of conformity.

- Number of retries:

Parameter code: 0000 0111

Length: 1 octet (0000 0001)

Parameter value: 0000 xxxx

The number of retries is represented by a binary number allowing a maximum of 16 retries coded in bits xxxx.

Value 0 of this parameter is illegal.

This parameter can only be present if b6(PH_RA) of the Class of conformity parameter is set to 1.

- “Cyclic flow” option:

Parameter code: 0000 1000

Length: 0 octet

d) Data

The data carried in this PDU is coded in a manner interpretable in the context of the application.

5.2.2.2.3 Association establishment response (AARP)

AARP structure is shown in Table 15.

Table 15 – Structure of an associated establishment response

Mapping	Meaning
---------	---------

octet 1	Size
octet 2	PDU code
octet 3	Calling AEI access point
octet 4	ditto
octet 5	ditto
octet 6	Called AEI access point
octet 7	ditto
octet 8	ditto
octet 9	Class of conformity
octet 10	Result
octet 11	Variable part
to	ditto
octet p	ditto
octet p+1	User data
to	ditto
end octet	ditto

a) Size

The minimum size value is 9 octets (coded 0000 1001), when the parameters relating to this PDU are not present and the maximum size value is reached when all the parameters are present.

b) Fixed part

The following fields are present in this part:

- The PDU code takes the value "1100 0000".
- The calling AEI access point, which corresponds to a link address (of ADAEI type) is represented with bit I/G as the high order bit in the lowest number octet.

Representation:

```

octet_3:    I / G x x x    x x x x
octet_4:    x  x x x      x x x x
octet_5:    S / R x x x    x x x x

```

- The called AEI access point which corresponds to a link address (of ADAEI type), is represented with the high order bit in the lowest number octet.

Representation:

```

octet_6:    I / G x x x    x x x x
octet_7:    x  x x x      x x x x
octet_8:    S / R x x x    x x x x

```

NOTE The default value used for this access point when the association establishment initiator does not wish to fill it in is the following value:

```

octet_6: 1111    1111
octet_7: 1111    1111
octet_8: 1111    1110

```

It is a link address value which is found in the generalised group address space reserved under the standard.

- The class of conformity

Representation:

```

octet_9:    b8 b7 b6 b5 b4 b3 b2 b1

```

Where:

```

b8 = 1, if PV_RE = 2
b7 = 1, if PH_AK = 1
b6 = 1, if PH_RA = 2
b5 = 1, if PH_SR = 2

```

b4 = 1, if PH_CF = 2

The possible combinations of these bits shall allow for the constraints on the classes of conformity.

- The result subsequent to an association establishment request is coded in one octet and indicated if the response is positive or negative, in the latter case providing information on the nature and location of the fault.

The values of the level 10 octet contain the following values in the following cases:

0000 0000: (+) Positive
0000 0001: (-) MCS provider (general)
0000 0010: (-) Responder user
0000 0011: (-) linked to "Cyclic rate"
0000 0100: (-) linked to "Transfer rate"
0000 0101: (-) linked to "Context name"
0000 0110: (-) linked to "Called entity identification"
0000 0111: (-) linked to "Calling entity identification"
0000 1000: (-) linked to "Service size"
xxxx xxxx: Spare

c) Variable part

The following parameters can be present in the variable part.

- Transfer rate:

The coding and semantics of this parameter are equivalent to those of the parameter of the same name in the association establishment request.

- Anticipation factor:

The coding and semantics of this parameter are equivalent to those of the parameter of the same name in the association establishment request PDU

- Application context:

The coding and semantics of this parameter are equivalent to those of the parameter of the same name in the association establishment request PDU (see 10.2.3.2.2.c).

- Called entity identification:

The coding and semantics of this parameter are equivalent to those of the parameter of the same name in the association establishment request PDU

- SDU size:

The coding and semantics of this parameter are equivalent to those of the parameter of the same name in the association establishment request PDU (see 10.2.3.2.2.c).

- PDU size:

The coding and semantics of this parameter are equivalent to those of the parameter of the same name in the association establishment request PDU (see 10.2.3.2.2.c).

- Number of retries:

The coding and semantics of this parameter are equivalent to those of the parameter of the same name in the association establishment request PDU (see 10.2.3.2.2.c).

d) Data

The data carried in this PDU are coded in a manner interpretable in the context of the application.

5.2.2.2.4 Association termination request (RERQ)

The RERQ structure is shown as in Table 16.

Table 16 – Structure of an association termination request

Mapping	Meaning
octet 1	Size
octet 2	PDU code
octet 3	User data
to	ditto
end octet	ditto

a) Length

The size value is 1 octet (coded 0000 0001).

b) Fixed part

The single field containing the fixed part is the PDU code which takes the value “1001 0000”.

c) Variable part

There is no variable part in this PDU.

d) Data

Data carried in this PDU has coding which is interpretable in the context of the application.

5.2.2.2.5 Association termination response (RERP)

The RERP structure is as in Table 17.

Table 17 – Structure of an association termination response

Mapping	Meaning
octet 1	Size
octet 2	PDU code
octet 3	Result
octet 4	User data
to	ditto
end octet	ditto

a) Size

The size value is 2 octets (coded: 0000 0010).

b) Fixed part

The fixed part of the header contains the following two fields:

- PDU code: “1101 0000”
- The result which is coded in one octet, indicating whether the result is positive or negative, and in the second phase providing information on the fault, to be defined later.

0000 0000: (+) Positive

0000 0001: (-) Negative

xxxx xxxx: Spare (Defaults)

c) Variable part

This PDU has no variable part.

d) Data

The data carried in this PDU has coding which is interpretable in the context of the application.

5.2.2.2.6 Association revocation request (ABRQ)

The ABRQ structure is as in Table 18.

Table 18 – Structure of an association revocation request

Mapping	Meaning
octet 1	Size
octet 2	PDU code
octet 3	Invocation origin
octet 4	User data
to	ditto
end octet	ditto

a) Size

The size value is 2 octets (coded: 0000 0010).

b) Fixed part

The following two fields make up the fixed part:

- PDU code taking value “1011 0000”.
- Invocation origin, coded in one octet and taking two values:

0000 0000: MCS user
1111 1111: MCS provider
xxxx xxxx: Other codes reserved

c) Variable part

This PDU has no variable part.

d) Data

The data carried in this PDU has coding which is interpretable in the context of the application.

5.2.2.2.7 Associated transfer request (ATRQ)

The ATRQ structure is as in Table 19.

Table 19 – Structure of an associated transfer request

Mapping	Meaning
octet 1	Size
octet 2	PDU code
octet 3	Sequence number
octet 4	ditto
octet 5	User data
to	ditto
end octet	ditto

a) Size

The size value is 3 octets (coded: 0000 0011).

b) Fixed part

The following two fields constitute the fixed part of the header:

- PDU code, taking values:

0000 0000: for unacknowledged transfers

0010 0XXX: for acknowledged transfers
where XXX is used to code the packet nature in the case of segmentation.

- The packet nature is coded as follows:

000: NIL (no segmentation)
100: STT (first packet)
101: MID (middle packet)
110: END (last packet)
111: ALL (first and last packet).

- The sequence number is represented in two octets by means of a binary number. This makes it possible to discriminate between 65536 data transfers. This number is not significant in the case of unacknowledged transfer.

c) Variable part

This PDU has no variable part.

d) Data

The data carried in this PDU have coding which is interpretable in the context of the application specific to the association.

5.2.2.2.8 Associated transfer acknowledgement (AKAT)

The AKAT structure is as in Table 20.

Table 20 – Structure of an associated transfer acknowledgement

Mapping	Meaning
octet 1	Size
octet 2	PDU code
octet 3	Sequence number
octet 4	ditto

a) Size

The size value is 3 octets (coded: 0000 0011).

b) Fixed part

The following field is contained in the fixed part:

- PDU number, taking the value "0110 0000".
- Sequence number, attributed by the numbering entity.

c) Variable part

This PDU has no variable part.

d) Data

This PDU has no user data.

5.2.2.2.9 Non-associated transfer request (NTRQ)

The NTRQ structure is as in Table 21.

Table 21 – Structure of a non-associated transfer request

Mapping	Meaning
---------	---------