

Industrial communication networks — Fieldbus specifications —

Part 6-14: Application layer protocol specification — Type 14 elements

ICS 25.040.40; 35.100.70

National foreword

This British Standard is the UK implementation of EN 61158-6-14:2008. It is identical with IEC 61158-6-14:2007. Together with all of the other sections of BS EN 61158-6, it supersedes BS EN 61158-6:2004 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications — Process measurement and control, including Fieldbus.

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

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

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

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 June 2008

© BSI 2008

ISBN 978 0 580 61634 1

Amendments/corrigenda issued since publication

Date	Comments

English version

**Industrial communication networks -
Fieldbus specifications -
Part 6-14: Application layer protocol specification -
Type 14 elements
(IEC 61158-6-14:2007)**

Réseaux de communication industriels -
Spécifications des bus de terrain -
Partie 6-14: Spécification des services
des couches d'application -
Éléments de type 14
(CEI 61158-6-14:2007)

Industrielle Kommunikationsnetze -
Feldbusse -
Teil 6-14: Protokollspezifikation
des Application Layer
(Anwendungsschicht) -
Typ 14-Elemente
(IEC 61158-6-14:2007)

This European Standard was approved by CENELEC on 2008-02-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

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

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

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

CENELEC

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

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Foreword

The text of document 65C/476/FDIS, future edition 1 of IEC 61158-6-14, prepared by SC 65C, Industrial networks, of IEC TC 65, Industrial-process measurement, control and automation, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61158-6-14 on 2008-02-01.

This and the other parts of the EN 61158-6 series supersede EN 61158-6:2004.

With respect to EN 61158-6:2004 the following changes were made:

- deletion of Type 6 fieldbus for lack of market relevance;
- addition of new fieldbus types;
- partition into multiple parts numbered 6-2, 6-3, ...6-20.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2008-11-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2011-02-01

NOTE Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in type combinations as specified explicitly in the EN 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61158-6-14:2007 was approved by CENELEC as a European Standard without any modification.

CONTENTS

INTRODUCTION.....	8
1 Scope.....	9
1.1 General.....	9
1.2 Specifications.....	9
1.3 Conformance.....	9
2 Normative references	10
3 Terms, definitions, symbols, abbreviations and conventions	10
3.1 Referenced terms and definitions	10
3.2 Fieldbus application layer specific terms and definitions	12
3.3 Abbreviations and symbols	14
3.4 Conventions	15
4 Abstract syntax.....	17
4.1 Fixed format PDU description	17
4.2 Object definitions in FAL management ASE.....	25
4.3 Definition of objects used in EPA application access entity.....	31
5 Transfer syntax.....	34
5.1 Encoding of basic data types	34
5.2 Encoding of EPA APDU header	40
5.3 Encoding of FAL management entity service parameters.....	40
5.4 Encoding of AAE Services	47
6 Structure of FAL protocol state machines	54
7 AP-Context state machine	54
7.1 Primitives exchanged between ALU and ALE.....	54
7.2 Protocol state machine descriptions	55
7.3 State transitions	55
7.4 Function descriptions	62
8 FAL management state machines	62
8.1 Primitives	62
8.2 Protocol state machine descriptions	64
8.3 State transitions	64
8.4 Function descriptions	66
9 Application access entity protocol machine.....	72
9.1 Primitives	72
9.2 AAE state machine	73
9.3 Event ASE protocol machine	76
9.4 Domain ASE protocol machine	77
10 Application relationship state machine.....	81
10.1 Primitives	81
10.2 AREP state description.....	83
10.3 State transitions	83
10.4 Function descriptions	84
11 DLL mapping protocol machine.....	85
11.1 Concept.....	85
11.2 Primitives	85
11.3 State description	85

11.4 State transitions	86
11.5 Function description	87
Annex ZA (normative) Normative references to international publications with their corresponding European publications.....	89
Bibliography.....	88
Figure 1 – State transition diagram	16
Figure 2 – Exchanged primitives of protocol state machine	54
Figure 3 – ACE protocol state machine	55
Figure 4 – FME protocol state machine	64
Figure 5 – AAE state transition diagrams	74
Figure 6 – Event ASE state transition diagrams	76
Figure 7 – Domain ASE state transition diagram	78
Figure 8 – AREP state transition diagrams.....	83
Figure 9 – ESME state transition.....	86
Table 1 – State machine description elements	16
Table 2 – Definition of EPA MOB header object	25
Table 3 – Definition of EPA device descriptor object	25
Table 4 – Definition of the time synchronization object.....	26
Table 5 – Definition of maximum response time object.....	27
Table 6 – Definition of the EPA communication scheduling management object.....	27
Table 7 – Definition of the device application information object	27
Table 8 – Definition of FB application information header.....	28
Table 9 – Definition of domain application information header.....	28
Table 10 – Definition of EPA link object header.....	29
Table 11 – Definition of FB application information object	29
Table 12 – Definition of EPA link object	30
Table 13 – Definition of domain application information object	30
Table 14 – Definition of domain object	31
Table 15 – Definition of simple variable object	32
Table 16 – Definition of event object	32
Table 17 – Definition of EPA socket mapping object	33
Table 18 – Definition of EPA socket timer object.....	33
Table 19 – Definition of ErrorType object	34
Table 20 – Encoding of Boolean value TRUE	34
Table 21 – Encoding of Boolean value FALSE	34
Table 22 – Encoding of Unsigned8 data type	34
Table 23 – Encoding of Unsigned16 data type	35
Table 24 – Encoding of Unsigned32 data type	35
Table 25 – Encoding of Unsigned64 data type	35
Table 26 – Encoding of Int8 data type	36
Table 27 – Encoding of Int16 data type	36
Table 28 – Encoding of Int32 data type	36

Table 29 – Encoding of Int64 data type	36
Table 30 – Encoding of Real type	37
Table 31 – Encoding of VisibleString data type	37
Table 32 – Encoding of OctetString data type	37
Table 33 – Encoding of BitString data type	38
Table 34 – Encoding of TimeOfDay data type	38
Table 35 – Encoding of BinaryDate data type.....	39
Table 36 – Encoding of TimeDifference data type	40
Table 37 – Encoding of EPA application layer service message header	40
Table 38 – Encoding of EM_DetectingDevice request parameters.....	41
Table 39 – Encoding of EM_OnlineReply request parameters	41
Table 40 – Encoding of EM_GetDeviceAttribute request parameters.....	42
Table 41 – Encoding of EM_GetDeviceAttribute positive response parameters	43
Table 42 – Encoding of EM_GetDeviceAttribute negative response parameters	44
Table 43 – Encoding of EM_ActiveNotification request parameters	44
Table 44 – Encoding of EM_ConfiguringDevice request parameters.....	45
Table 45 – Encoding of EM_ConfiguringDevice positive response parameters	46
Table 46 – Encoding of EM_ConfiguringDevice negative response parameters	46
Table 47 – Encoding of EM_SetDefaultValue request parameters	47
Table 48 – Encoding of EM_SetDefaultValue positive response parameters	47
Table 49 – Encoding of clear device attribute service refuse packet.....	47
Table 50 – Encoding of DomainDownload request parameters	48
Table 51 – Encoding of domain download service response packet	48
Table 52 – Encoding of DomainDownload negative response parameters	48
Table 53 – Encoding of DomainUpload request parameters	49
Table 54 – Encoding of DomainUpload positive response parameters.....	49
Table 55 – Encoding of DomainUpload negative response parameters	49
Table 56 – Encoding of EventReport request parameters.....	50
Table 57 – Encoding of EventReportAcknowledge request parameters	50
Table 58 – Encoding of EventReportAcknowledge positive response parameters.....	50
Table 59 – Encoding of EventReportAcknowledge negative response parameters	51
Table 60 – Encoding of ReportConditionChanging request parameters	51
Table 61 – Encoding of ReportConditionChanging positive response parameters.....	51
Table 62 – Encoding of ReportConditionChanging negative response parameters	51
Table 63 – Encoding of Read request parameters.....	52
Table 64 – Encoding of Read positive response parameters	52
Table 65 – Encoding of Read negative response parameters	52
Table 66 – Encoding of Write request parameters	53
Table 67 – Encoding of Write positive response parameters	53
Table 68 – Encoding of Write negative response parameters	53
Table 69 – Encoding of VariableDistribute request parameters	53
Table 70 – Primitives delivered by ALU to ALE	54
Table 71 – Primitives delivered by ALE to ALU	55

Table 72 – ACE state descriptions	55
Table 73 – ACE state transitions (sender).....	56
Table 74 – ACE state transitions (receiver).....	59
Table 75 – APServiceType() descriptions.....	62
Table 76 – Primitives delivered by application layer user to FME	62
Table 77 – Primitives delivered by FME to application layer user	63
Table 78 – Primitive parameters exchanged between FME and application layer user.....	63
Table 79 – Primitives delivered by FME to ESME.....	63
Table 80 – Primitives delivered by ESME to FME.....	63
Table 81 – Primitives parameters exchanged between FME and ESME	64
Table 82 – State transitions of EPA FME	65
Table 83 – EpaRcvNewIpAddress() descriptions	67
Table 84 – EpaAttribute_Set() descriptions	67
Table 85 – EpaRestoreDefaults() descriptions	67
Table 86 – EpaNewAddress() descriptions	67
Table 87 – Restart_EPAREpeatTimer() descriptions.....	68
Table 88 – EpaClear_DuplicatePdTagFlag() descriptions.....	68
Table 89 – EPAREpeatTimerExpire() descriptions	68
Table 90 – EpaSend_EM_ReqRspMessage() descriptions	69
Table 91 – EpaSend_EM_CommonErrorRsp() descriptions	69
Table 92 – EpaSntpSyncLost() descriptions	69
Table 93 – EpaIPAddressCollision() descriptions	69
Table 94 – EpaRecvMsg() descriptions	70
Table 95 – EpaQueryMatch() descriptions.....	70
Table 96 – EpaMessageIDMatch() descriptions.....	70
Table 97 – EpaDevId_Match() descriptions	71
Table 98 – EpaPdTag_Match() descriptions	71
Table 99 – EpaSet_Attribute_Data() descriptions	71
Table 100 – EpaSet_DuplicatePdTagFlag() descriptions	71
Table 101 – Primitives issued by ALU to AAE	72
Table 102 – Primitives issued by AAE to ALU	72
Table 103 – Primitives parameters exchanged between AAE and ALU.....	72
Table 104 – Primitives issued by AAE to ESME	73
Table 105 – Primitives issued by ESME to AAE	73
Table 106 – Primitive parameters exchanged between AAE and ESME	73
Table 107 – AAE state descriptions	73
Table 108 – AAE state transitions (sender)	74
Table 109 – AAE state transitions (receiver)	75
Table 110 – ServiceType() descriptions	76
Table 111 – State value of event management.....	76
Table 112 – Event ASE state transition table	77
Table 113 – Domain state value.....	77
Table 114 – Domain ASE state transition table	78

Table 115 – Domain_DownloadSucceed() description	80
Table 116 – Domain_WriteBuffer() description	81
Table 117 – IncreamentInvokeDomainCounter() description.....	81
Table 118 – DecreamentInvokeDomainCounter() description	81
Table 119 – Primitives issued by FME(or AAE) to AREP	82
Table 120 – Primitives issued by AREP to FME(or AAE)	82
Table 121 – Primitives parameters exchanged between AREP and FME(or AAE)	82
Table 122 – Primitives issued by AREP to ESME	82
Table 123 – Primitives issued by ESME to AREP	82
Table 124 – Primitive parameters exchanged between AREP and ESME	83
Table 125 – AREP state descriptions	83
Table 126 – AREP state transitions.....	83
Table 127 – AREPType() descriptions.....	84
Table 128 – ServiceType() descriptions	85
Table 129 – The primitives exchanged between transport layer and ESME	85
Table 130 – Primitives parameters exchanged between Transport Layer and ESME	86
Table 131 – ESME state description	86
Table 132 – ECFME state transitions	87
Table 133 – ServiceType()description	87

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 6-14: Application layer protocol specification – Type 14 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 14 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 14 fieldbus application layer in terms of

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

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

- 1) define the wire-representation of the service primitives defined in IEC 61158-5-14, and
- 2) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 14 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 61158-5-14.

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

1.3 Conformance

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

Conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

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

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

IEC 61158-3-14, *Industrial communication networks – Fieldbus specifications – Part 3-14: Data-link layer service definition – Type 14 elements*

IEC 61158-4-14, *Industrial communication networks – Fieldbus specifications – Part 4-14: Data-link layer protocol specification – Type 14 elements*

IEC 61158-5-14, *Industrial communication networks – Fieldbus specifications – Part 5-14: Application layer service definition – Type 14 elements*

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

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

ISO/IEC 8802-3, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

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

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

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 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 7498-1 apply:

- a) application entity
- b) application process
- c) application protocol data unit
- d) application service element
- e) application entity invocation

- f) application process invocation
- g) application transaction
- h) real open system
- i) transfer syntax

3.1.2 ISO/IEC 8822 terms

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

- a) abstract syntax
- b) presentation context

3.1.3 ISO/IEC 9545 terms

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

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

3.1.4 ISO/IEC 8824 terms

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

- a) object identifier
- b) type

3.1.5 Fieldbus data-link Layer terms

For the purposes of this document, the following terms as defined in IEC 61158-3-14 and IEC 61158-4-14 apply.

- a) DL-Time
- b) DL-Scheduling-policy
- c) DLCEP
- d) DLC
- e) DL-connection-oriented mode
- f) DLPDU
- g) DLSDU
- h) DLSAP
- i) link
- j) network address
- k) node address
- l) node
- m) scheduled

3.2 Fieldbus application layer specific terms and definitions

3.2.1

access control

control on the reading and writing of an object

3.2.2

access Path

association of a symbolic name with a variable for the purpose of open communication

3.2.3

communication macrocycle

set of basic cycles needed for a configured communication activity in a macro network segment

3.2.4

communication scheduling

algorithms and operation for data transfers occurring in a deterministic and repeatable manner

3.2.5

configuration (of a system or device)

step in system design: selecting functional units, assigning their locations and defining their interconnections.

3.2.6

cyclic

repetitive in a regular manner

3.2.7

destination FB Instance

FB instance that receives the specified parameters

3.2.8

domain

part of memory used to store code or data

3.2.9

domain download

operation to write data in a domain

3.2.10

domain upload

operation to read data from a domain

3.2.11

entity

particular thing, such as a person, place, process, object, concept, association, or event

3.2.12

Type 14 bridge

DL-relay entity which performs synchronization between links (buses) and may perform selective store-and-forward and routing functions to connect two micro network segments

3.2.13

identifier

16-bit word associated with a system variable

3.2.14

index

address of an object within an application process

3.2.15

instance

actual physical occurrence of an object within a class that identifies one of many objects within the same object class

3.2.16

instantiation

creation of an instance of a specified type.

3.2.17

management information

network-visible information for the purpose of managing the field system.

3.2.18 +

management information base

organized list of management information

3.2.19

mapping

set of values having defined correspondence with the quantities or values of another set.

3.2.20

member

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

3.2.21

message filtering

decision on a message according to a special rule.

3.2.22

micro segment

part of a network, where special scheduling is implemented.

3.2.23

offset

number of octets from a specially designated position.

3.2.24

phase

elapsed fraction of a cycle, measured from some fixed origin

3.2.25

process interface

data exchange and information mapping between physical process and application unit

3.2.26

real-time

ability of a system to provide a required result in a bounded time

3.2.27

real-time communication

transfer of data in real-time.

3.2.28**Real-Time Ethernet (RTE)**

ISO/IEC 8802-3-based network that includes real-time communication

NOTE 1 Other communication can be supported, providing the real-time communication is not compromised.

NOTE 2 This definition is dedicated, but not limited, to ISO/IEC 8802-3. It could be applicable to other IEEE 802 specifications, for example IEEE 802.11.

3.2.29**schedule**

temporal arrangement of a number of related operations.

3.2.30**scheduling macrocycle**

time interval to implement a specific schedule.

3.2.31**source FB Instance**

FB instance that sends a specific parameter

3.2.32**time offset**

time difference from a specially designated time

3.3 Abbreviations and symbols

AAE	Application Access Entity
AE	Application Entity
AL	Application Layer
ALE	Application Layer Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
ARP	Address Resolution Protocol
AREP	Application Relationship End Point
ASE	Application Service Element
Cnf	Confirmation
CR	Communication Relationship
CREP	Communication Relationship End Point
CSMA/CD	Carrier Sense Multiple Access Protocol with Collision Detection
DD	Device Description
DHCP	Dynamic Host Configuration Protocol
DL-	(as a prefix) data-link-
DLCEP	Data-link Connection End Point
DLL	Data-link Layer
DLE	Data-link Entity
DLM	Data-link-management
DLS	Data-link Service
DLSAP	Data-link Service Access Point
DLSDU	DL-service-data-unit

ECSME	Type 14 communication scheduling management entity
Type 14	Ethernet for Plant Automation
EM_	(as a prefix) Type 14 Management
ESME	Type 14 Socket Mapping Entity
FB	Function Block
FBAP	Function Block Application Process
FME	FAL Management Entity
Ind	Indication
IP	Internet Protocol
LLC	Logical Link Control
LMP	Link Management Protocol
MAC	Medium Access Control
MAU	Medium Attachment Unit
MOB	Management Object Base
PAD	Pad (bits)
PDU	Protocol Data Unit
P/S	Publisher/Subscriber
Req	Request
Rsp	Response
RTE	Real-Time Ethernet
RT-Ethernet	Real-Time Ethernet
SAP	Service Access Point
SDU	Service Data Unit
SME	System Management Entity
SNTP	Simple Network Time Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
.cnf	Confirm Primitive
.ind	Indication Primitive
.req	Request Primitive
.rsp	Response Primitive

3.4 Conventions

3.4.1 General concept

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of three parts: its class definitions, its services, and its protocol specification. The first two are contained in IEC 61158-5-14. 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 61158-5-14. The service specification defines the services that are provided by the ASE.

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

3.4.2 Conventions for state machines for Type 14

A state machine describes the state sequence of an entity and can be represented by a state transition diagram and/or a state table.

In a state transition diagram (Figure 1), the transition between two states represented by circles is illustrated by an arrow beside which the transition events or conditions are presented.

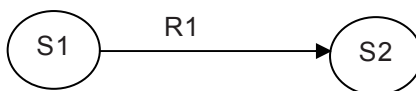


Figure 1 – State transition diagram

Table 1 – State machine description elements

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

The conventions used in the state transition table (Table 1) are as follows.

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

xxx A parameter name.

Example:

Identifier := reason

means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'
"xxx" Indicates fixed value.

Example:

Identifier := "abc"

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

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

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

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

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

&& Logical "AND"

|| Logical "OR"

Service.req represents a Request Primitive; Service.req{} indicates that a request primitive is sent;

Service.ind represents an Indication Primitive; Service.ind{} indicates that an Indication Primitive is received;

Service.rsp represents a Response Primitive; Service.rsp{} indicates that a Response Primitive is sent;

Service.cnf represents a Confirm Primitive; Service.cnf{} indicates that a Confirm Primitive is received.

4 Abstract syntax

4.1 Fixed format PDU description

EPA PDU consists of fixed-length PDU header and variable-length PDU body. The former contains service type, message type and message length, etc.

```
EPAPDU ::= CHOICE {
    confirmed-RequestPDU          [0]    IMPLICIT Confirmed-RequestPDU,
    confirmed-ResponsePDU        [1]    IMPLICIT Confirmed-ResponsePDU,
    confirmed-ErrorPDU           [2]    IMPLICIT Confirmed-ErrorPDU,
    unconfirmed-RequestPDU       [3]    IMPLICIT Unconfirmed-RequestPDU
}
```

```
Confirmed-RequestPDU ::= SEQUENCE {
    pduHeader          PDUHeader,
    confirmed-request  Confirmed-Request
}
```

```
Confirmed-ResponsePDU ::= SEQUENCE {
    pduHeader          PDUHeader,
    confirmed-response Confirmed-Response
}
```

```
Confirmed-ErrorPDU ::= SEQUENCE {
    pduHeader          PDUHeader,
    confirmed-error    Confirmed-Error
}
```

```
Unconfirmed-RequestPDU ::= SEQUENCE {
    pduHeader          PDUHeader,
    unconfirmed-request Unconfirmed-Request
}
```

4.1.1 Confirmed request service

```
Confirmed- Request ::= CHOICE {
    EM_GetDeviceAttribute [0]    IMPLICIT EM_GetDeviceAttribute-RequestPDU,
    EM_ConfiguringDevice [1]    IMPLICIT EM_ConfiguringDevice-RequestPDU,
    EM_SetDefaultValue [2]    IMPLICIT EM_SetDefaultValue-RequestPDU,
    DomainDownload [3]    IMPLICIT DomainDownload-RequestPDU,
    DomainUpload [4]    IMPLICIT DomainUpload-RequestPDU,
    AcknowledgeEventRoport [5]    IMPLICIT AcknowledgeEventNotifi-RequestPDU,
    ReportConditionChanging [6]    IMPLICIT AlterEventConditionMon-RequestPDU,
    Read [7]    IMPLICIT Read-RequestPDU,
    Write [8]    IMPLICIT Write-RequesPDU
}
```

4.1.2 Confirmed response service

```
Confirmed- Response ::= CHOICE {
    EM_GetDeviceAttribute [0] IMPLICIT EM_GetDeviceAttribute-ResponsePDU,
    EM_ConfiguringDevice [1] IMPLICIT EM_ConfiguringDevice-ResponsePDU,
    EM_SetDefaultValue [2] IMPLICIT EM_SetDefaultValue-ResponsePDU,
    DomainDownload [3] IMPLICIT DomainDownload-ResponsePDU,
    DomainUpload [4] IMPLICIT DomainUpload-ResponsePDU,
    AcknowledgeEventRoport [5] IMPLICIT AcknowledgeEventNotifi-ResponsePDU,
    ReportConditionChanging [6] IMPLICIT AlterEventConditionMon-ResponsePDU,
    Read [7] IMPLICIT Read-ResponsePDU,
    Write [8] IMPLICIT Write-ResponsePDU
}
```

}

4.1.3 Confirmed error

Confirmed- Error : : = CHOICE {

EM_GetDeviceAttribute	[0]	IMPLICIT Error-Type,
EM_ConfiguringDevice	[1]	IMPLICIT Error-Type,
EM_SetDefaultValue	[2]	IMPLICIT Error-Type,
DomainDownload	[3]	IMPLICIT Error-Type,
DomainUpload	[4]	IMPLICIT Error-Type,
AcknowledgeEventReport	[5]	IMPLICIT Error-Type,
ReportConditionChanging	[6]	IMPLICIT Error-Type,
Read	[7]	IMPLICIT Error-Type,
Write	[8]	IMPLICIT Error-Type

}

4.1.4 Error type

ErrorType : : = SEQUENCE {

ErrorClass	[0]	IMPLICIT Integer8,
ErrorCode	[1]	IMPLICIT Integer8,
AdditionalCode	[2]	IMPLICIT Integer8,
Reserved	[3]	IMPLICIT OctetString,
AdditionalDescription	[4]	IMPLICIT VisibleString

}

4.1.5 Error class

ErrorClass ::= CHOICE {		ErrorCode
Resource	[0] IMPLICIT Integer8 {	
	memory-unavailable	(0),
	Other	(1)
}		
Service	[1] IMPLICIT Integer8 {	
	object-state-conflict	(0),
	object-constraint-conflict	(1),
	parameter-inconsistent	(2),
	illegal-parameter	(3),
	Size Error	(4),
	Other	(5)
}		
Access	[2] IMPLICIT Integer8 {	
	object-access-unsupported	(0),
	object-non-existent	(1),
	object-access-denied	(2),
	hardware-fault	(3),
	type-conflict	(4),
	object-attribute-inconsistent	(5),
	Access-to-element-unsupported	(6),
	Other	(7)
}		
Timer	[3] IMPLICIT Integer8 {	
	Timer-Expire	(0),
	Timer-Error	(1),
	Other	(2)
}		
Other	[4] IMPLICIT Integer8 {	
	Other	(0)
}		
}		

4.1.6 Unconfirmed request

Unconfirmed-Request: : = CHOICE {

EM_DetectingDevice	[0]	IMPLICIT EM_DetectingDevice-RequestPDU,
EM_OnlineReply	[1]	IMPLICIT EM_OnlineReply-RequestPDU,
EM_ActiveNotification	[2]	IMPLICIT EM_ActiveNotification-RequestPDU,

EventReport	[3]	IMPLICIT	EventReport-RequestPDU,
VariableDistribute	[4]	IMPLICIT	VariableDistribute-RequestPDU

}

4.1.7 EPA application layer PDU

ApplicationLayerPDU ::= SEQUENCE {
 PDUHeader,
 PDUBody CHOICE {
 Confirmed-Request,
 Confirmed-Response,
 Confirmed-Error,
 Unconfirmed-Request
 }
}

4.1.8 APDU header format

PDUHeader ::= SEQUENCE {
 ServiceID [0] IMPLICIT Unsigned8,
 Reserved [1] IMPLICIT OctetString,
 Length [2] IMPLICIT Unsigned16,
 MessageID [3] IMPLICIT Unsigned16
}

4.1.9 FAL Management Entity services

4.1.9.1 EM_DetectingDevice service

EM_DetectingDevice-RequestPDU ::= SEQUENCE {
 QueryType [0] IMPLICIT Unsigned8,
 Reserved [1] IMPLICIT OctetString,
 PDTag [2] IMPLICIT VisibleString,
 FBTag [3] IMPLICIT VisibleString,
 ElementID [4] IMPLICIT Unsigned16
}

4.1.9.2 EM_OnlineReply service

EM_OnlineReply -RequestPDU ::= SEQUENCE {
 QueryType [0] IMPLICIT Unsigned8,
 DuplicateTagDetected [1] IMPLICIT Boolean,
 Reserved [2] IMPLICIT OctetString,
 QueriedObjectIpAddress [3] IMPLICIT Unsigned32,
 QueriedObjectDeviceID [4] IMPLICIT VisibleString,
 QueriedObjectPDTag [5] IMPLICIT VisibleString
}

4.1.9.3 EM_GetDeviceAttribute service

EM_GetDeviceAttribute-RequestPDU ::= SEQUENCE {
 DestinationIPAddress [0] IMPLICIT Unsigned32,
}

EM_GetDeviceAttribute-ResponsePDU ::= CHOICE {
 EM_GetDeviceAttribute-PositiveResponsePDU,
 EM_GetDeviceAttribute-NegativeResponsePDU
}

EM_GetDeviceAttribute-PositiveResponsePDU ::= SEQUENCE {
 DeviceID [0] IMPLICIT VisibleString,
 PdTag [1] IMPLICIT VisibleString,
 Status [2] IMPLICIT Unsigned8,
 DeviceType [3] IMPLICIT Unsigned8,
}

Annunciation Interval	[4]	IMPLICIT	Unsigned16,
Annunciation Version Number	[5]	IMPLICIT	Unsigned16,
DuplicateTagDetected	[6]	IMPLICIT	Boolean,
DeviceRedundancyNumber	[7]	IMPLICIT	Unsigned8,
LANRedundancyPort	[8]	IMPLICIT	Unsigned16
DeviceRedundancy State	[9]	IMPLICIT	Unsigned8,
MaxRedundancyNumber	[10]	IMPLICIT	Unsigned8,
ActiveIPAddress	[11]	IMPLICIT	Unsigned32

}

EM_GetDeviceAttribute-NegativeResponsePDU ::= SEQUENCE {

DestinationIPAddress	[0]	IMPLICIT	Unsigned32,
Error-Type	[1]	IMPLICIT	Error-Type

}

4.1.9.4 EM_ActiveNotification service

EM_ActiveNotification-RequestPDU ::= SEQUENCE {

DeviceID	[0]	IMPLICIT	VisibleString,
PdTag	[1]	IMPLICIT	VisibleString,
Status	[2]	IMPLICIT	Unsigned8,
DeviceType	[3]	IMPLICIT	Unsigned8,
AnnunciationVersionNumber	[4]	IMPLICIT	Unsigned16,
Device Redundancy Number	[5]	IMPLICIT	Unsigned8,
DeviceRedundancyState	[6]	IMPLICIT	Unsigned8,
LANRedundancyPort	[7]	IMPLICIT	Unsigned16
DuplicateTagDetected	[8]	IMPLICIT	Boolean,
MaxRedundancyNumber	[9]	IMPLICIT	Unsigned8,
Reserved	[10]	IMPLICIT	OctetString,
ActiveIPAddress	[11]	IMPLICIT	Unsigned32

}

4.1.9.5 EM_ConfiguringDevice service

EM_ConfiguringDevice-RequestPDU ::= SEQUENCE {

DestinationIPAddress	[0]	IMPLICIT	Unsigned32,
DeviceID	[1]	IMPLICIT	VisibleString,
PdTag	[2]	IMPLICIT	VisibleString,
AnnunciationInterval	[3]	IMPLICIT	Unsigned16,
DuplicateTagDetected	[4]	IMPLICIT	Boolean,
DeviceRedundancyNumber	[5]	IMPLICIT	Unsigned8,
LANRedunancyPort	[6]	IMPLICIT	Unsigned16,
DeviceRedundancyState	[7]	IMPLICIT	Unsigned8,
MaxRedundancyNumber	[8]	IMPLICIT	Unsigned8,
ActiveIPAddress	[9]	IMPLICIT	Unsigned32

}

EM_ConfiguringDevice-ResponsePDU ::= CHOICE {

EM_ConfiguringDevice-PositiveResponsePDU,

EM_ConfiguringDevice-NegativeResponsePDU

}

EM_ConfiguringDevice-PositiveResponsePDU ::= SEQUENCE {

DestinationIPAddress	[0]	IMPLICIT	Unsigned32,
MaxRedundancyNumber	[1]	IMPLICIT	Unsigned8

}

EM_ConfiguringDevice-NegativeResponsePDU ::= SEQUENCE {

DestinationIPAddress	[0]	IMPLICIT	Unsigned32,
ErrorType	[1]	IMPLICIT	ErrorType

}

4.1.9.6 EM_SetDefaultValue service

```

EM_SetDefaultValue-RequestPDU ::= SEQUENCE {
    DestinationIPAddress      [0]    IMPLICIT Unsigned32,
    DeviceID                  [1]    IMPLICIT VisibleString,
    PdTag                     [2]    IMPLICIT VisibleString
}

EM_SetDefaultValue-ResponsePDU ::= CHOICE {
    EM_SetDefaultValue-PositiveResponsePDU,
    EM_SetDefaultValue-NegativeResponsePDU
}

EM_SetDefaultValue-PositiveResponsePDU ::= SEQUENCE {
    DestinationIPAddress      [0]    IMPLICIT Unsigned32
}

EM_SetDefaultValue-NegativeResponsePDU ::= SEQUENCE {
    DestinationIPAddress      [0]    IMPLICIT Unsigned32,
    ErrorType                 [1]    IMPLICIT ErrorType
}
    
```

4.1.10 Application Access Entity (AAE) services

4.1.10.1 DomainDownload service

```

DomainDownload-RequestPDU ::= SEQUENCE {
    SourceAppID               [0]    IMPLICIT Unsigned16,
    DestinationAppID          [1]    IMPLICIT Unsigned16,
    DestinationObjectID       [2]    IMPLICIT Unsigned16,
    DataNumber                [3]    IMPLICIT Unsigned16,
    MoreFollows               [4]    IMPLICIT Boolean,
    Reserved                  [5]    IMPLICIT OctetString,
    DataLength                [6]    IMPLICIT Unsigned16,
    LoadData                 [7]    IMPLICIT OctetString
}

DomainDownload-ResponsePDU ::= CHOICE {
    DomainDownload-PositiveResponsePDU,
    DomainDownload-NegativeResponsePDU
}

DomainDownload-PositiveResponsePDU ::= SEQUENCE {
    DestinationAppID          [0]    IMPLICIT Unsigned16
}

DomainDownload-NegativeResponsePDU ::= SEQUENCE {
    DestinationAppID          [0]    IMPLICIT Unsigned16,
    Reserved                  [1]    IMPLICIT OctetString,
    ErrorType                 [2]    IMPLICIT ErrorType
}
    
```

4.1.10.2 DomainUpload service

```

DomainUpload-RequestPDU ::= SEQUENCE {
    SourceAppID               [0]    IMPLICIT Unsigned16,
    DestinationAppID          [1]    IMPLICIT Unsigned16,
    DestinationObjectID       [2]    IMPLICIT Unsigned16,
    DataNumber                [3]    IMPLICIT Unsigned16
}
    
```

```

DomainUpload-ResponsePDU ::= CHOICE {
    DomainUpload-PositiveResponsePDU,
    DomainUpload-NegativeResponsePDU
}

```

```

DomainUpload-PositiveResponsePDU ::= SEQUENCE {
    DestinationAppID          [0] IMPLICIT Unsigned16,
    DataLength                [1] IMPLICIT Unsigned16,
    MoreFollows               [2] IMPLICIT Boolean,
    Reserved                  [3] IMPLICIT OctetString,
    LoadData                 [4] IMPLICIT OctetString
}

```

```

DomainUpload-NegativeResponsePDU ::= SEQUENCE {
    DestinationAppID          [0] IMPLICIT Unsigned16,
    Reserved                  [1] IMPLICIT OctetString,
    ErrorType                 [2] IMPLICIT ErrorType
}

```

4.1.10.3 EventRoport service

```

EventRoport-RequestPDU ::= SEQUENCE {
    DestinationAppID          [0] IMPLICIT Unsigned16,
    SourceAppID               [1] IMPLICIT Unsigned16,
    SourceObjectID            [2] IMPLICIT Unsigned16,
    EventNumber               [3] IMPLICIT Unsigned16,
    EventData                 [4] IMPLICIT OctetString
}

```

4.1.10.4 AcknowledgeEventRoport service

```

AcknowledgeEventRoport-RequestPDU ::= SEQUENCE {
    DestinationAppID          [0] IMPLICIT Unsigned16,
    DestinationObjectID       [1] IMPLICIT Unsigned16,
    EventNumber               [2] IMPLICIT Unsigned16
}

```

```

AcknowledgeEventRoport -ResponsePDU ::= CHOICE {
    AcknowledgeEventRoport -PositiveResponsePDU,
    AcknowledgeEventRoport -NegativeResponsePDU
}

```

```

AcknowledgeEventRoport -PositiveResponsePDU: ::= SEQUENCE{
    DestinationAppID          [0] IMPLICIT Unsigned16
}

```

```

AcknowledgeEventRoport -NegativeResponsePDU: ::= SEQUENCE{
    DestinationAppID          [0] IMPLICIT Unsigned16,
    Reserved                  [1] IMPLICIT OctetString,
    ErrorType                 [2] IMPLICIT ErrorType
}

```

4.1.10.5 ReportConditionChanging service

```

ReportConditionChanging-RequestPDU ::= SEQUENCE {
    DestinationAppID          [0] IMPLICIT Unsigned16,
    DestinationObjectID       [1] IMPLICIT Unsigned16,
    Enabled                   [2] IMPLICIT Boolean
}

```

```

ReportConditionChanging -ResponsePDU ::= CHOICE {

```

ReportConditionChanging -PositiveResponsePDU,
ReportConditionChanging -NegativeResponsePDU

}

ReportConditionChanging -PositiveResponsePDU: : = SEQUENCE{

DestinationAppID [0] IMPLICIT Unsigned16

}

ReportConditionChanging -NegativeResponsePDU: : = SEQUENCE{

DestinationAppID [0] IMPLICIT Unsigned16

Reserved [1] IMPLICIT OctetString,

ErrorType [2] IMPLICIT ErrorType

}

4.1.10.6 Read service

Read-RequestPDU : : = SEQUENCE {

DestinationAppID [0] IMPLICIT Unsigned16,

DestinationObjectID [1] IMPLICIT Unsigned16,

SubIndex [2] IMPLICIT Unsigned16

}

Read -ResponsePDU : : = CHOICE {

Read -PositiveResponsePDU,

Read -NegativeResponsePDU

}

Read-PositiveResponsePDU : : = SEQUENCE {

DestinationAppID [0] IMPLICIT Unsigned16,

Reserved [1] IMPLICIT OctetString,

Data [2] IMPLICIT OctetString

}

Read-NegativeResponsePDU : : = SEQUENCE {

DestinationAppID [0] IMPLICIT Unsigned16,

Reserved [1] IMPLICIT OctetString,

ErrorType [2] IMPLICIT ErrorType

}

4.1.10.7 Write service

Write-RequestPDU : : = SEQUENCE {

DestinationAppID [0] IMPLICIT Unsigned16,

DestinationObjectID [1] IMPLICIT Unsigned16,

SubIndex [2] IMPLICIT Unsigned16,

Reserved [3] IMPLICIT OctetString,

Data [4] IMPLICIT OctetString

}

Write -ResponsePDU : : = CHOICE {

Write -PositiveResponsePDU,

Write -NegativeResponsePDU

}

Write -PositiveResponsePDU : : = SEQUENCE {

DestinationAppID [0] IMPLICIT Unsigned16,

}

Write -NegativeResponsePDU : : = SEQUENCE {

DestinationAppID	[0]	IMPLICIT	Unsigned16,
Reserved	[1]	IMPLICIT	OctetString,
ErrorType	[2]	IMPLICIT	ErrorType

}

4.1.10.8 VariableDistribute service

VariableDistribute-RequestPDU : : = SEQUENCE {

SourceAppID	[0]	IMPLICIT	Unsigned16,
SourceObjectID	[1]	IMPLICIT	Unsigned16,
Data	[2]	IMPLICIT	OctetString

}

4.1.11 Abstract syntax of data type

4.1.11.1 Notation of Boolean type

Boolean ::= BOOLEAN

--value is non-zero means TRUE.

--value is zero means FALSE.

4.1.11.2 Notation of integer type

Int8 ::= INTEGER (-128..+127) -- integer range $-2^7 \leq i \leq 2^7-1$

Int16 ::= INTEGER (-32768..+32767) -- integer range $-2^{15} \leq i \leq 2^{15}-1$

Int32 ::= INTEGER -- integer range $-2^{31} \leq i \leq 2^{31}-1$

Int64 ::= INTEGER -- integer range $-2^{63} \leq i \leq 2^{63}-1$

4.1.11.3 Notation of unsigned integer type

Unsigned8 ::= INTEGER (0..255) -- integer range $0 \leq i \leq 2^8-1$

Unsigned16 ::= INTEGER (0..65535) -- integer range $0 \leq i \leq 2^{16}-1$

Unsigned32 ::= INTEGER -- integer range $0 \leq i \leq 2^{32}-1$

Unsigned64 ::= INTEGER -- integer range $0 \leq i \leq 2^{64}-1$

4.1.11.4 Notation of float data type

Real ::= BIT STRING SIZE (4) -- IEC-60559 single precision

4.1.11.5 Notation of visible string type

VisibleString ::= VISIBLE STRING --general use

4.1.11.6 Notation of octet string type

OctetString ::= Octet STRING --general use

4.1.11.7 Notation of bit string type

BitString ::= BIT STRING -- general use

4.1.11.8 Notation of TimeOfDay type

TimeOfDay ::= OctetString6

4.1.11.9 Notation of binary date type

BinaryDate ::= OctetString8

4.1.11.10 Notation of TimeDifference type

TimeDifference ::= OctetString6

4.2 Object definitions in FAL management ASE

4.2.1 EPA MOB Header

The object of the EPA MOB Header is defined as shown in Table 2.

Table 2 – Definition of EPA MOB header object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of EPA MOB header object in the EPA MOB
2	MOB Revision Number	Read Only	Unsigned16	2	2	The version of EPA MOB

4.2.2 EPA device descriptor object

The object of the EPA Device Descriptor Object is defined as shown in Table 3.

Table 3 – Definition of EPA device descriptor object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of EPA Device Descriptor Object in the MOB
2	Reserved	Read Only	OctetString	2	2	Reserved
3	Device ID	Read Only	VisibleString	4	32	Device ID
4	PD_Tag	Read Only	VisibleString	36	32	Device Tag
5	Active IP Address	Read Only	Unsigned32	68	4	Current operational IP address
6	Device Type	Read Only	Unsigned8	72	1	Device type
7	Status	Read Only	Unsigned8	73	1	Device status
8	Device Version	Read Only	Unsigned16	74	2	Device version number
9	Annunciation Interval	Read Only	Unsigned16	76	2	The interval of devices broadcast its annunciation
10	Annunciation Version Number	Read Only	Unsigned16	78	2	Annunciation version number of devices broadcast
11	Device Redundancy State	Read Only	Unsigned8	80	1	Device redundancy status
12	Device Redundancy Number	Read Only	Unsigned8	81	1	Device redundancy number

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
13	LANRedundancyPort	Read Only	Unsigned16	82	2	Redundant messages processing port of the device
14	Max Redundancy Number	Read Only	Unsigned8	84	1	Maximum redundancy number of the device
15	Duplicate Tag Detected	Read Only	Boolean	85	1	This property describes whether the device's PD_Tag is in collision with another device

4.2.3 Time synchronization object

The object of the Time Synchronization Object class is defined as shown in Table 4:

Table 4 – Definition of the time synchronization object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of the Time Synchronization Object in the MOB
2	Reserved	Read Only	OctetString	2	2	Reserved
3	Primary Time Server	Read/Write	Unsigned32	4	4	IP address of master time server
4	Secondary Time Server	Read/Write	Unsigned32	8	4	IP address of slave time server
5	Time Request Timeout	Read Only	Unsigned32	12	4	The maximum time that time client waits for the response of time server in seconds
6	Time Request Interval	Read/Write	Unsigned32	16	4	The time interval that time client requests the time server
7	Capable Time Sync Class	Read Only	Unsigned32	20	4	The synchronization precision supported by time client
8	Target Time Sync Class	Read/Write	Unsigned32	24	4	The required synchronization precision as to the time client
9	Current Time	Read Only	BinaryDate	28	8	Current time of the device
10	Standard Time Difference	Read Only	TimeDifference	36	6	Standard time difference

4.2.4 Maximum response time object

The object of the Maximum Response Time Object class is defined as shown in Table 5.

Table 5 – Definition of maximum response time object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of object in the MOB
2	Reserved	Read Only	OctetString	2	2	Reserved
3	Max Response Time	Read/Write	TimeDifference	4	4	The maximum response time of confirmed service in milliseconds

4.2.5 EPA communication scheduling management object

The object of the EPA Communication Scheduling Management class is defined as shown in Table 6.

Table 6 – Definition of the EPA communication scheduling management object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of object in the MOB
2	Reserved	Read Only	OctetString	2	2	Reserved
3	Communication MacroCycle	Read/Write	TimeDifference	4	4	The communication macro period of subnet which the device belongs to
4	NonPeriodic Data Transfer Offset	Read/Write	TimeDifference	8	4	The offset of non-periodic message begin to transmit relative to the start of communication macro period
5	Communication Macrocycle Version Number	Read Only	Unsigned16	12	2	The version number of communication macro period

4.2.6 Device application information object

The object of the Device Application Information class is defined as shown in Table 7.

Table 7 – Definition of the device application information object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of object in the MOB
2	XDDL Version	Read Only	Unsigned16	2	2	The device description version number

4.2.7 FB application information header

The object of the Encoding of FB Application Information Header class is defined as shown in Table 8.