IEC 61158-6-14

Edition 1.0 2007-12

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 6-14: Application layer protocol specification – Type 14 elements

IEC 61158-6-14:2007(E)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61158-6-14

Edition 1.0 2007-12

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 6-14: Application layer protocol specification – Type 14 elements

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE **XC**

ICS 35.100.70; 25.040.40

ISBN 2-8318-9489-1

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as “IEC Publication(s)”). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

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

International Standard IEC 61158-6-14 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This first edition and its companion parts of the IEC 61158-6 subclauses cancel and replace IEC 61158-4:2003. This edition of this part constitutes a technical addition. This part and its Type 14 companion parts also cancel and replace IEC/PAS 62409, published in 2005.

This edition of IEC 61158-6 includes the following significant changes from the previous edition:

a) deletion of the former Type 6 fieldbus for lack of market relevance;
b) addition of new types of fieldbuses;

c) partition of part 6 of the third edition into multiple parts numbered -6-2, -6-3, …

The text of this standard is based on the following documents:

FDIS	Report on voting
65C/476/FDIS	65C/487/RVD

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

This publication has been drafted in accordance with ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under http://webstore.iec.ch in the data related to the specific publication. At this date, the publication will be:

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

NOTE The revision of this standard will be synchronized with the other parts of the IEC 61158 series.

The list of all the parts of the IEC 61158 series, under the general title *Industrial communication networks – Fieldbus specifications*, can be found on the IEC web site.

INTRODUCTION

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

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-14: 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.

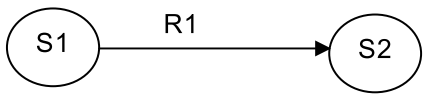

Figure 1 – State transition diagram

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,
        AcknowledgeEventRoport       [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,
```

```
        EventRoport                 [3]   IMPLICIT  EventRoport-RequestPDU,
        VariableDistribute          [4]   IMPLICIT  VariableDistribute-RequestPDU
    }
```

4.1.7 EPA application layer PDU

```
ApplicationLayerPDU ::=  SEQUENCE {
        PDUHeader,
        PDUBody              CHOICE {
                Confirmed-Request,
                Confirmed-Response,
                Confirmed-Error,
                Uniconfirmed-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,
        DuplicateTagDetacted        [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<= i <= 2^7-1

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

Int32 ::= INTEGER -- integer range -2^{31}<= i <= 2^{31}-1

Int64 ::= INTEGER -- integer range -2^{63}<= i <= 2^{63}-1

4.1.11.3 Notation of unsigned integer type

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

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

Unsigned32 ::= INTEGER -- integer range 0 <= i <= 2^{32}-1

Unsigned64 ::= INTEGER -- integer range 0 <= i <= 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.

Table 8 – Definition of FB application information header

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	Number of FB Application Information Object	Read Only	Unsigned16	2	2	The number of FB Application Information Object
3	First Number of FB Application Information Object	Read Only	Unsigned16	4	2	First Number of FB Application Information Object

4.2.8 Domain application information header

The object of the Domain Application Information Header class is defined as shown in Table 9.

Table 9 – Definition of domain application information header

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of Domain Application Information Header object in the MOB
2	Number of Domain Application Information Object	Read Only	Unsigned16	2	2	Number of Domain Application Information Objects in the device
3	First Number of Domain Application Object	Read Only	Unsigned16	4	2	First Number of Domain Application Object in the MOB
4	Number of Configured Domain Object	Read Only	Unsigned16	6	2	Number of Configured Domain Objects
5	Number of UnConfigured Domain Object	Read Only	Unsigned16	8	2	Number of UnConfigured Domain Objects

4.2.9 EPA link object header

The object of the EPA Link Object Header class is defined as shown in Table 10.

Table 10 – Definition of EPA link object header

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	The index of Link Object Header t in the MOB
2	Number of Link Object	Read Only	Unsigned16	2	2	Number of Link Object in the device
3	First Number of Link Object	Read Only	Unsigned16	4	2	First Number of Link Object in the MOB
4	Number of Configured Link Object	Read Only	Unsigned16	6	2	Number of Configured Link Objects
5	Number of UnConfigured Link Object	Read Only	Unsigned16	8	2	Number of UnConfigured Link Objects

4.2.10 FB application information object

The object of the FB Application Information class is defined as shown in Table 11.

Table 11 – Definition of FB 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 FB Application Information Object in the MOB
2	Reserved	Read Only	Unsigned16	2	2	Reserved
3	FB Name	Read Only	VisibleString	4	32	FB Name, its length is 32 octets, if the string length is less than 32 octets, then the remained part are padded with BLANK(0x20)
4	FB Type	Read Only	Unsigned16	36	2	FB Type
5	Max Number of Instantiation	Read Only	Unsigned16	38	2	The maximum instances number of FB
6	FB Execution Time	Read Only	Unsigned32	40	4	The execution time of FB in milliseconds
7	First Number of Instantiation	Read Only	Unsigned16	44	2	First Instance Number allocated to FB instance when it is instantiated

4.2.11 EPA link object

The object of the EPA Link Object class is defined as shown in Table 12.

Table 12 – Definition of EPA link 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 Link Object in the MOB
2	LocalAppID	Read/Write	Unsigned16	2	2	Instance ID of local instance
3	Local Object ID	Read/Write	Unsigned16	4	2	The index of local variable object
4	RemoteAppID	Read/Write	Unsigned16	6	2	Instance ID of remote FB
5	RemoteObjectID	Read/Write	Unsigned16	8	2	ID of remote element object
6	ServiceOperation	Read/Write	Unsigned8	10	1	EPA service ID used by Link Object
7	ServiceRole	Read/Write	Unsigned8	11	1	Role of local object in the communication process
8	RemoteIPAddress	Read/Write	Unsigned32	12	4	IP address of remote device; if local and destination FB instance objects are in the same EPA device, then this property can be ignored; if the EPA service use the broadcast or multicast method, then this property should be broadcast or multicast group address
9	SendTimeOffset	Read/Write	TimeDifference	16	4	Time offset when sending periodic packet from the start time of a communication macrocycle. Its data type is 4 octets of TimeDifference

4.2.12 Domain application information object

The object of the Domain Application Information class is defined as shown inTable 13.

Table 13 – Definition of domain 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 the domain application information object in the MOB
2	Domain Object ID	Read Only	Unsigned16	2	2	The index of the domain object corresponding to the domain application information object
3	ConfigurationStatus	Read Only	Boolen	4	1	The configuration status of domain object, Boolean type, if its value is true, then it shows that the domain object is not configured
4	Reserved	Read Only	OctetString	5	3	Reserved
5	Domain Name	Read Only	Unsigned16	8	32	The name of the domain object, its length is 32 octets, the unused part is padded with BLANK (0x20)

4.3 Definition of objects used in EPA application access entity

This subclause defines the encodings of objects in EPA application access entity.

4.3.1 Domain object

The object of the Domain class is defined as shown in Table 14.

Table 14 – Definition of domain object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	ObjectID	Read Only	Unsigned16	2	The index of Domain Object
2	Domain Name	Read/Write	VisibleString	32	The name of Domain Object
3	Max Octets	Read Only	Unsigned16	2	The maximum octets in the domain
4	Password	Read/Write	Unsigned16	2	The password used to access the Domain Object
5	AccessGroups	Read/Write	Unsigned8	1	The access group of Domain Object
6	AccessRights	Read/Write	Unsigned8	1	The access right of Domain Object
7	Local Address	Read Only	Unsigned32	4	The pointer pointed to the specific Domain Object. If not used, its value should be set to 0xFFFFFFFF
8	Domain State	Read Only	Unsigned8	1	The status of domain object, it can be the following value: 0 ——EXISTENT 1 ——DOWNLOADING 2 ——UPLOADING 3 ——READY 4 ——IN-USE
9	Last State	Read Only	Unsigned8	1	The status of domain object before upload/download, the meaning of its value if shown as follows: 0 ——EXISTENT 1 ——DOWNLOADING 2 ——UPLOADING 3 ——READY 4 ——IN-USE
10	Used Application Counter	Read Only	Unsigned16	2	The number of programs using the domain now, if the counter value is bigger than 0, it shows that this domain is being used, so it cannot be overwritten by the download service, its data type is unsigned16

4.3.2 Simple variable object

The definition of Simple Variable Object is shown in Table 15.

Table 15 – Definition of simple variable object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	ObjectID	Read Only	Unsigned16	2	The index of the Variable Object in the MOB
2	Data Type	Read Only	Unsigned8	1	The data type of the Variable Object.
3	Length	Read Only	Unsigned16	2	The length of Variable Object in octet
4	Local Address	Read Only	Unsigned32	4	The pointer pointed to the specific Variable Object which can be used to internally address the Domain Object. If not used , its value should be set to 0xFFFFFFFF
5	Password	Read/Write	Unsigned16	2	The password used to access the Variable Object.
6	AccessGroups	Read/Write	Unsigned8	1	The access group of Variable Object
7	AccessRights	Read/Write	Unsigned8	1	The access right to Variable Object

4.3.3 Event object

The definition of Event Object is shown in Table 16.

Table 16 – Definition of event object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	ObjectID	Read Only	Unsigned16	2	The ID of the Event Object
2	Length	Read Only	Unsigned16	2	The octet length of the Event Object
3	Password	Read/Write	Unsigned16	2	The password used to access the Domain Object.
4	AccessGroups	Read/Write	Unsigned8	1	The access group of the Domain Object
5	AccessRights	Read/Write	Unsigned8	1	The access right of the Domain Object
6	Local Address	Read Only	Unsigned32	4	The pointer pointed to the specific Event Object, it is used to internally address the Variable Object. If not used , its value should be set to 0xFFFFFFFF
7	Enabled	Read Only	Boolean	1	Enabled = TRUE ⇔ UNLOCKED Signifies the event object isn't locked, and the event can be sent out. Enabled = FALSE ⇔ LOCKED Signifies the event object is locked, and the event can not be sent out

4.3.4 EPA socket mapping object

The definition of EPA Socket Mapping Object is shown in Table 17.

Table 17 – Definition of EPA socket mapping object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	LocalIPAddress	Read Only	Unsigned32	4	IP address of local device
2	RemoteIPAddress	Read Only	Unsigned32	4	IP address of remote device
3	ActiveUdpPort	Read Only	Unsigned16	2	The UDP port used when sending message
4	ActiveServiceID	Read Only	Unsigned16	2	Service ID
5	ActiveMesssagLength	Read Only	Unsigned16	2	The length of the message waiting to be sent
6	ActiveMessageID	Read Only	Unsigned16	2	Active packet ID, i.e. the MessageID
7	ActiveMesssageTime	Read Only	Time Difference	6	The response time of the active message, this parameter shows the maximum response time of the active message, if it does not need the response, it should be set to zero. The unit is millisecond
8	ActiveDataPointer	Read Only	Unsigned32	4	The pointer to the header of the active message
9	MaxMessageLength	Read Only	Unsigned16	2	The maximum message length allowed. If the user level message exceeds the length, it will be denied to send, and will return an error flag
10	MaxRetransmitNumber	Read Only	Unsigned16	2	The maximum retransmission times allowed

4.3.5 EPA socket timer object

The definition of the EPA Socket Timer Object is shown in Table 18.

Table 18 – Definition of EPA socket timer object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	TimerID	Read Only	Unsigned16	2	The ID of the timer
2	ActiveServiceID	Read Only	Unsigned16	2	Service ID which indicates the service used to send the message
3	ActiveMessageID	Read Only	Unsigned16	2	The active message ID, i.e. Message ID in the message
4	ActiveMessageTime	Read Only	Unsigned32	4	The response time of the active message, this parameter shows the maximum response time of the active message, if it does not need the response, it should be set to zero. The unit is millisecond

4.3.6 ErrorType object

The definition of the ErrorType Object is shown in Table 19.

Table 19 – Definition of ErrorType object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	Error Class	Read Only	Int8	1	Error class
2	Error Code	Read Only	Int8	1	Error code, this parameter gives a more concrete error description
3	Additional Code	Read Only	Int8	1	Additional code, this parameter is optional, and the user can define its usage according to its own need
4	Reserved	Read Only	Octetstring	1	Reserved
5	Additional Description	Read Only	VisibleString	32	Additional description, this parameter is optional, and it used to add a text description in the error information. Its data type is VisibleString

5 Transfer syntax

5.1 Encodingof basic data types

5.1.1 Boolean

A Boolean value is encoded in one octet, all bits are set to 0 for FALSE and 1 for TRUE.

If the value is TRUE, the value of each bit is shown in the Table 20 (Bit 7 is MSB, Bit 0 is LSB).

Table 20 – Encoding of Boolean value TRUE

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	1	1	1	1	1	1	1	1

If the value is FALSE, the value of each bit is shown in Table 21.

Table 21 – Encoding of Boolean value FALSE

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	0	0	0	0	0	0	0	0

5.1.2 Unsigned8

The Unsigned8 type is encoded in one octet, the range is 0~255, the weight of each bit is shown in Table 22.

Table 22 – Encoding of Unsigned8 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

5.1.3 Unsigned16

The Unsigned16 type is encoded in two octets, the range is 0~216-1, the weight of each bit is shown in Table 23.

Table 23 – Encoding of Unsigned16 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.4 Unsigned32

The Unsigned32 type is encoded in four octets, the range is 0~232-1, the weight of each bit is shown in Table 24.

Table 24 – Encoding of Unsigned32 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octet2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octet3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.5 Unsigned64

The Unsigned64 type is encoded in eight octets, the range is 0~264-1, the weight of each bit is shown in Table 25.

Table 25 – Encoding of Unsigned64 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Octet2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Octet3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
Octet4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Octet5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octet6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octet7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.6 Int8

The Int8 type is encoded in one octet, the range is -128~127. If the sign bit (SN) is 1, the data is negative, otherwise, the data is positive or zero when bit SN is 0. The weight of each bit is shown in Table 26.

Table 26 – Encoding of Int8 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	SN	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.7 Int16

The Int16 type is encoded in two octets, the range is $-2^{15}\sim2^{15}-1$. If bit SN is 1, the data is negative; otherwise, the data is positive or zero when bit SN is 0. The weight of each bit is shown in Table 27.

Table 27 – Encoding of Int16 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.8 Int32

The Int32 type is encoded in four octets, the range is $-2^{31}\sim2^{31}-1$. If the SN is 1, the data is negative; otherwise, the data is positive or zero when bit SN is 0. The weight of each bit is shown in Table 28.

Table 28 – Encoding of Int32 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octet2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octet3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.9 Int64

The Int64 type is encoded in eight octets, the range is $-2^{63}\sim2^{63}-1$. If the value of bit SN is 1, the data is negative; otherwise, the data is positive or zero when bit SN is 0. The weight of each bit is shown in Table 29.

Table 29 – Encoding of Int64 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Octet2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Octet3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
Octet4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Octet5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octet6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octet7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.10 Real

The Real type is encoded in four octets. Its range is in conformance with that of IEEE Std 754 Short Real Number (total 32 bits). If the value of bit SN is 1, the data is negative, otherwise, the data is positive or zero when bit SN is 0. Bits 6 to 0 of octet 1 and bit 7 of octet 2 defines the exponent field. It is followed by the fraction field from bit 6 of octet 1 to bit 0 of octet 2. The weight of each bit is shown in Table 30.

Table 30 – Encoding of Real type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
Octet2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Octet3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
Octet4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

5.1.11 VisibleString

The VisibleString type is encoded in visible string, the length is variable. The definition is in conformance with that of ISO 646 and ISO 2375. The encoding is shown in Table 31.

Table 31 – Encoding of VisibleString data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	the first character							
Octet2	the second character							
……	and so on……							
……	so on……							
Octetn								

5.1.12 OctetString

The OctetString type is encoded in one or several octets which are aligned from 1 to n according to the sequence number. The encoding is shown in Table 32.

Table 32 – Encoding of OctetString data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	Binary data							
Octet2	Binary data							
……	……							
……	……							
Octetn								

5.1.13 BitString

The BitString type is encoded in a group of Octets, the length is variable from 1 to n, n is an arbitrary natural number. The encoding is shown in Table 33.

Table 33 – Encoding of BitString data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	0	1	2	3	4	5	6	7
Octet2	8	9	10	11	12	13	14	15
……	so on……							
……	so on……							
Octetn								

5.1.14 TimeOfDay

The TimeOfDay type consists of a date and a time, it is encoded in total 6 octets. The date field is encoded as Unsigned16 data (octet 1 and octet 2), it is stated in days relatively to the first of January 2000. On the first of January 2000, the date starts with the value zero. The time is stated in milliseconds since midnight, at midnight the counting starts with the value zero. The time field is encoded as an Unsigned32 data (from octet 3 to octet 6). The encoding is shown in Table 34.

```
{
    Unsigned16  Date;          //days
    Unsigned32  Millisecond;   //milliseconds
}
```

Table 34 – Encoding of TimeOfDay data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Octet3	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octet4	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octet5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.15 BinaryDate

The type BinaryDate consists of a calendar date and a time, it is encoded in 8 octets shown in Table 35.

The year field is encoded as an Unsigned16 data (octet1 and octet2), if its value is 2004, it represents the year of 2004.

The month field is encoded as an Unsigned8 data (octet 3), the range is 1~12, its value represents one of 12 months in a year.

The date field is encoded as an Unsigned8 data (octet 4), the range is 1~31, its value represents one day of 31 days in a month.

The hour field is encoded as an Unsigned8 data (octet5), the range is 0~23, its value represents 1 h of 24 h in a day.

The minute field is encoded as an Unsigned8 data (octet 6), the range is 0~59, its value represents 1 min of 60 min in an hour.

The millisecond field is encoded as an Unsigned16 data (octet7 and octet 8), the range is 0~59999, its value represents 1 ms of 60 000 ms in a minute.

```
{
    Unsigned16  Year;           //year
    Unsigned8   Month;          //month
    Unsigned8   Date;           //day
    Unsigned8   Hour;           //hour
    Unsigned8   Minute;         //minute
    Unsigned16  Millisecond;    //millisecond
}
```

Table 35 – Encoding of BinaryDate data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Octet3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Octet4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Octet5	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Octet6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Octet7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.16 TimeDifference

The type TimeDifference consists of a date and a time, it is encoded in 4 or 6 octets ; its value states the time difference.

The selective date field is encoded as an Unsigned16 data (octet 1 and octet 2), its value represents the days difference.

The time field is stated in milliseconds and encoded as an Unsigned32 data (from octet 3 to octet 6), its value represents the milliseconds difference. The encoding is shown inTable 36.

```
{
    Unsigned16  Date;           //days difference
    Unsigned32  Millisecond;    // the fraction part of a day expressed in milliseconds
}
```

Table 36 – Encoding of TimeDifference data type

	B7	B6	B5	B4	B3	B2	B1	B0
Octet1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Octet3	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octet4	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octet5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octet6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.17 Encoding of SEQUENCE

The SEQUENCE structure is comparable with a record. It represents a collection of user data of the same or of different Data Types.

A structure may contain a simple variable or further structures as components.

5.1.18 Encoding of CHOICE

A CHOICE represents a selection from a set of predefined possibilities.

5.2 Encoding of EPA APDU header

The encoding of the EPA Application layer Service Message packet header is shown in Table 37.

Table 37 – Encoding of EPA application layer service message header

No.	Parameter Name	Data type	Octet offset	Octet length	Description
1	ServiceID	Unsigned8	0	1	This parameter describes the service type and message type. Bit 7 to 6 indicates the message type: 00 – request message 01 – response message 10 – error message 11 – reserved The lowest six bits used to signify the service ID
2	Reserved	OctetString	1	3	Reserved
3	Length	Unsigned16	4	2	This parameter describes the length of the whole message
4	MessageID	Unsigned16	6	2	This parameter describes the ID of the message

5.3 Encoding of FAL management entity service parameters

5.3.1 EM_DetectingDevice service

The EM_DetectingDevice request parameters are coded as shown in Table 38.

Table 38 – Encoding of EM_DetectingDevice request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Query Type	Unsigned8	0	1	Signifies the query request type: 0: Query according to PD_Tag. The following parameter is PD_Tag. 1: Query according to FB Tag. The following parameter is FB Tag. 2: Query according to ElementID. The following parameter is FB Tag and ElementID
2	Reserved	Octetstring	1	3	Reserved
3	PD_Tag	VisibleString	4	32	The physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20).
4	FB Tag	VisibleString	36	32	Function block instance tag which is used to query the information of EPA device which include the FB instance
5	Element ID	Unsigned16	68	2	Element ID in FB which must be used with FB Tag together, because ElementID is unique only in one FB instance

5.3.2 EM_OnlineReply service

The EM_OnlineReply request parameters are coded as shown in Table 39.

Table 39 – Encoding of EM_OnlineReply request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Query Type	Unsigned8	0	1	Signifies the Query Type: 0: Query according to PD_Tag. 1: Query according to FB Tag. 2: Query according to ElementID
2	Duplicate Tag Detected	Boolean	1	1	This property describes if the device's PD_Tag collides with the other devices' PD_Tag (i.e. repeated PD_Tag). TRUE= PD_Tags Collide
3	Reserved	Octetstring	2	2	Reserved
4	Queried Object IP Address	Unsigned32	4	4	IP address of the queried EPA physical device (i.e. the IP address of the local device)
5	Queried Object Device ID	VisibleString	8	32	The queried EPA physical device ID (e.g. local device ID), its length is 32 octets, and the unused part is padded with BLANK (0x20)
6	Queried Object PD_Tag	VisibleString	40	32	The queried EPA physical device tag (e.g. local device tag). Its length is 32 octets, and the unused part is padded with BLANK (0x20)

5.3.3 EM_GetDeviceAttribute service

5.3.3.1 Request primitive

The EM_GetDeviceAttribute request parameters are coded as shown in Table 40.

Table 40 – Encoding of EM_GetDeviceAttribute request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Destination IP Addres	Unsigned32	0	4	IP address of the target device

5.3.3.2 Positive response primitive

The EM_GetDeviceAttribute positive response parameters are coded as shown in Table 41.

Table 41 – Encoding of EM_GetDeviceAttribute positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device ID	VisibleString	0	32	Local device ID
2	PD_Tag	VisibleString	32	32	The physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)
3	Status	Unsigned8	64	1	The status of the EPA device: 0: no address 1: unconfigured 2: configured, operational
4	Device Type	Unsigned8	65	1	Local device type
5	Annunciation Interval	Unsigned16	66	2	The interval of the annunciation message sent out by the device
6	Annunciation Version Number	Unsigned16	68	2	The version number of the annunciation message
7	Duplicate Tag Detected	Boolean	70	1	This property indicates whether PD_Tag of the device is in collision with the other or not (i.e. duplicated PD_Tag). TRUE= PD_Tag in collision
8	Redundancy Number	Unsigned8	71	1	The redundancy number of local device, if the device is active, then this value is zero, and has no following parameters
9	Device Redundancy State	Unsigned8	72	1	The redundancy status the local device is in: 0——active status 1——back-up status If the redundancy number is 0, then the response primitive does not contain this parameter
10	Max Redundancy Number	Unsigned8	73	1	The maximum redundancy number of the device, if the redundancy number is 0, then the response primitive does not contain this parameter
11	Reserved	Octetstring	74	2	Reserved
12	Active IP Address	Unsigned32	76	4	IP address of the active device (if no redundancy, then its local IP address); if the redundancy number is 0, then the response primitive does not contain this parameter

5.3.3.3 Negative response primitive

The EM_GetDeviceAttribute negative response parameters are coded as shown in Table 42.

Table 42 – Encoding of EM_GetDeviceAttribute negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationIPAddress	Unsigned32	0	4	IP address of the target device
2	Error Type	ErrorType	4	N	See ErrorType

5.3.4 EM_ActiveNotification service

The EM_ActiveNotification request parameters are coded as shown in Table 43.

Table 43 – Encoding of EM_ActiveNotification request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device ID	VisibleString	0	32	ID of local device
2	PD_Tag	VisibleString	32	32	The local physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)
3	Status	Unsigned8	64	1	The status of the EPA device: 0——no address; 1——unconfigured; 2——configured,operational
4	Device Type	Unsigned8	65	1	Local device type
5	Annunciation version number	Unsigned16	66	2	The version number of the annunciation message
6	Device Redundancy Number	Unsigned8	68	1	The redundancy number of local device, if it is anactive device, then this value is 0, otherwise the following parameter is invalid
7	Device Redundancy State	Unsigned8	69	1	The redundancy status of the local device: 0——active state 1——backup state If the redundancy number is 0, then the response primitive does not contain this parameter
8	LAN Redundancy Port	Unsigned16	70	2	The LAN redundancy message processing port of the device which requests the service

No.	Parameter name	Data type	Octet offset	Octet length	Description
9	Duplicate Tag Detected	Boolean	72	1	This property describes if the device's PD_Tag collides with the other devices' PD_Tag (i.e. duplicated PD_Tag). TRUE= PD_Tags Collide
10	Reserved	Octetstring	73	2	Reserved
11	Max. Redundancy Number	Unsigned8	75	1	The maximum redundancy number of the device
12	Active IP Address	Unsigned32	76	4	IP address of the active device (if no redundancy, then its local IP address); if the redundancy number is 0, then the response primitive does not contain this parameter

5.3.5 EM_ConfiguringDevice service

5.3.5.1 Request Primitive

The EM_ConfiguringDevice request parameters are coded as shown in Table 44.

Table 44 – Encoding of EM_ConfiguringDevice request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationIPAddress	Unsigned32	0	4	Destination IP Address
2	Device ID	VisibleString	4	32	ID of local device
3	PD_Tag	VisibleString	36	32	The local physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)
4	Annunciation Interval	Unsigned16	68	2	The interval of the annunciation message sent out by the device
5	Duplicate Tag Detected	Boolean	70	1	This property describes if the device's PD_Tag collides with the other devices' PD_Tag (i.e. duplicated PD_Tag). TRUE= PD_Tags Collide
6	Device Redundancy Number	Unsigned8	71	1	The redundancy number of the local device, if its active device, then this value is 0, and does not have the following parameters
7	LAN Redundancy Port	Unsigned16	72	2	The LAN redundancy message processing port of the device which requests the service

No.	Parameter name	Data type	Octet offset	Octet length	Description
8	Device Redundancy State	Unsigned8	74	1	The redundancy status of the local device: 0——active state 1——backup state If the redundancy number is 0, then the response primitive does not contain this parameter
9	Max Redundancy Number	Unsigned8	75	1	The maximum redundancy number of the device
10	Active IP Address	Unsigned32	80	4	IP address of the active device (if no redundancy, then its local IP address); if the redundancy number is 0, then the response primitive does not contain this parameter

5.3.5.2 Positive response primitive

The EM_ConfiguringDevice positive response parameters are coded as shown in Table 45.

Table 45 – Encoding of EM_ConfiguringDevice positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Destination IP Address	Unsigned32	0	4	Destination IP Address
2	Max Redundancy Number	Unsigned8	4	1	The maximum redundancy number of the device, if the redundancy number is 0, the response primitive does not contain this parameter

5.3.5.3 Negative response primitive

The EM_ConfiguringDevice negative response parameters are coded as shown in Table 46.

Table 46 – Encoding of EM_ConfiguringDevice negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Destination IP Addres	Unsigned32	0	4	IP address of the target device
2	Error Type	ErrorType	4	N	See Error Type

5.3.6 EM_SetDefaultValue service

5.3.6.1 Request primitive

The EM_SetDefaultValue request parameters are coded as shown in Table 47.

Table 47 – Encoding of EM_SetDefaultValue request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationIPAddress	Unsigned32	0	4	Destination IP address
2	Device ID	VisibleString	4	32	ID of the local device
3	PD_Tag	VisibleString	36	32	The local physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)

5.3.6.2 Positive response primitive

The EM_SetDefaultValue positive response parameters are coded as shown in Table 48.

Table 48 – Encoding of EM_SetDefaultValue positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Destination IP Address	Unsigned32	0	4	Destination IP address

5.3.6.3 Negative response primitive

The EM_SetDefaultValue negative response parameters are coded as shown in Table 49.

Table 49 – Encoding of clear device attribute service refuse packet

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Destination IP Address	Unsigned32	0	4	Destination IP address
2	Error Type	ErrorType	4	N	See Error Type

5.4 Encoding of AAE Services

5.4.1 DomainDownload Service

5.4.1.1 Request primitive

The DomainDownload request parameters are coded as shown in Table 50.

Table 50 – Encoding of DomainDownload request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	SourceAppID	Unsigned16	0	2	Application ID of the source device
2	DestinationAppID	Unsigned16	2	2	Application ID of destination device
3	DestinationObjectID	Unsigned16	4	2	Domain object ID of the destination device
4	DataNumber	Unsigned16	6	2	Download times
5	MoreFollows	Boolean	8	1	Flag used to signify if there are more downloads following
6	Reserved	OctetSring	9	1	Reserved
7	DataLength	Unsigned16	10	2	Data length, The range is 0~512
8	Load Data	OctetString	12	N	Domain download data

5.4.1.2 Positive response primitive

The DomainDownload positive response parameters are coded as shown in Table 51.

Table 51 – Encoding of domain download service response packet

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Destination IP address

5.4.1.3 Negative response primitive

DomainDownload negative response parameters are coded as shown in Table 52.

Table 52 – Encoding of DomainDownload negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Destination IP address
2	Reserved	Octetstring	2	2	Reserved
3	ErrorType	ErrorType	4	N	See Error Type

5.4.2 Domain Upload Service

5.4.2.1 Request primitive

The DomainUpload request parameters are coded as shown in Table 53.

Table 53 – Encoding of DomainUpload request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	SourceAppID	Unsigned16	0	2	Application ID of the source device
2	DestinationAppID	Unsigned16	2	2	Application ID of destination device
3	DestinationObjectID	Unsigned16	4	2	Domain object ID of the destination device
4	DataNumber	Unsigned16	6	2	Download times

5.4.2.2 Positive response primitive

The DomainUpload positive response parameters are coded as shown in Table 54.

Table 54 – Encoding of DomainUpload positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	DataLength	Unsigned16	2	2	Data length, the ranget is 0~512
3	MoreFollows	Boolean	4	1	Flag used to indicate whether there are more downloads data
4	Reserved	Octetstring	5	3	Reserved
5	Load Data	Octetstring	8	N	Domain upload data

5.4.2.3 Negative response primitive

The DomainUpload negative response parameters are coded as shown in Table 55.

Table 55 – Encoding of DomainUpload negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	Reserved	Octetstring	2	2	Reserved
3	Error Type	ErrorType	4	N	See Error Type

5.4.3 EventRoport Service

The EventRoport request parameters are coded as shown in Table 56.

Table 56 – Encoding of EventRoport request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	SourceAppID	Unsigned16	2	2	Application ID of source device
3	SourceObjectID	Unsigned16	4	2	Domain object ID of source device
4	EventNumber	Unsigned16	6	2	Number of the event
5	EventData	OctetString	8	N	Specific event data

5.4.4 EventRoportAcknowledge Service

5.4.4.1 Request primitive

The EventRoportAcknowledge request parameters are coded as shown in Table 57.

Table 57 – Encoding of EventRoportAcknowledge request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	DestinationObjectID	Unsigned16	2	2	Object ID of destination device
3	EventNumber	Unsigned16	4	2	Event number

5.4.4.2 Positive response primitive

The EventRoportAcknowledge positive response parameters are coded as shown in Table 58.

Table 58 – Encoding of EventRoportAcknowledge positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device

5.4.4.3 Negative response primitive

The EventRoportAcknowledge negative response parameters are coded as shown in Table 59.

Table 59 – Encoding of EventRoportAcknowledge negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	Reserved	Octetstring	2	2	Reserved
3	Error Type	ErrorType	4	N	See Error Type

5.4.5 ReportConditionChanging service

5.4.5.1 Request primitive

The ReportConditionChanging request parameters are coded as shown in Table 60.

Table 60 – Encoding of ReportConditionChanging request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	DestinationObjectID	Unsigned16	2	2	Object ID of destination device
3	Enabled	Boolean	4	1	Enable flag
4	Reserved	Octetstring	5	3	Reserved

5.4.5.2 Positive response primitive

The ReportConditionChanging positive response parameters are coded as shown in Table 61.

Table 61 – Encoding of ReportConditionChanging positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device

5.4.5.3 Negative response primitive

The ReportConditionChanging negative response parameters are coded as shown in Table 62.

Table 62 – Encoding of ReportConditionChanging negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	Reserved	Octetstring	2	2	Reserved
3	Error Type	ErrorType	4	N	See Error Type

5.4.6 Read service

5.4.6.1 Request primitive

The Read request parameters are coded as shown in Table 63.

Table 63 – Encoding of Read request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	DestinationObjectID	Unsigned16	2	2	Object ID of destination device
3	SubIndex	Unsigned16	4	2	SubIndex of the accessed object

5.4.6.2 Positive response primitive

The Read positive response parameters are coded as shown in Table 64.

Table 64 – Encoding of Read positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	Reserved	Octetstring	2	2	Reserved
3	Data	Octetstring	4	N	Returned data

5.4.6.3 Negative response primitive

The Read negative response parameters are coded as shown in Table 65.

Table 65 – Encoding of Read negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	Reserved	Octetstring	2	2	Reserved
3	Error Type	ErrorType	4	N	See Error Type

5.4.7 Write Service

5.4.7.1 Request primitive

The Write request parameters are coded as shown in Table 66.

Table 66 – Encoding of Write request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	DestinationObjectID	Unsigned16	2	2	Object ID of destination device
3	SubIndex	Unsigned16	4	2	SubIndex of the written object
4	Reserved	Octetstring	6	2	Reserved
5	Data	Octetstring	8	N	Data written

5.4.7.2 Positive response primitive

The Write positive response parameters are coded as shown in Table 67.

Table 67 – Encoding of Write positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device

5.4.7.3 Negative response primitive

The Write negative response parameters are coded as shown in Table 68.

Table 68 – Encoding of Write negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationAppID	Unsigned16	0	2	Application ID of destination device
2	Reserved	Octetstring	2	2	Reserved
3	Error Type	ErrorType	4	N	See Error Type

5.4.8 VariableDistribute Service

The VariableDistribute request parameters are coded as shown in Table 69.

Table 69 – Encoding of VariableDistribute request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	SourceAppID	Unsigned16	0	2	Application ID of the source device
2	SourceObjectID	Unsigned16	2	2	Object ID of the source device
3	Data	Octetstring	4	N	VariableDistributed data

6 Structure of FAL protocol state machines

Interface to FAL services and protocol machines are specified in this subclause. Conventions used for the descriptions are given in the generic part of this standard.

Figure 2 illustrates the relationships of primitives. The primitives exchanged between each other will be described as follows.

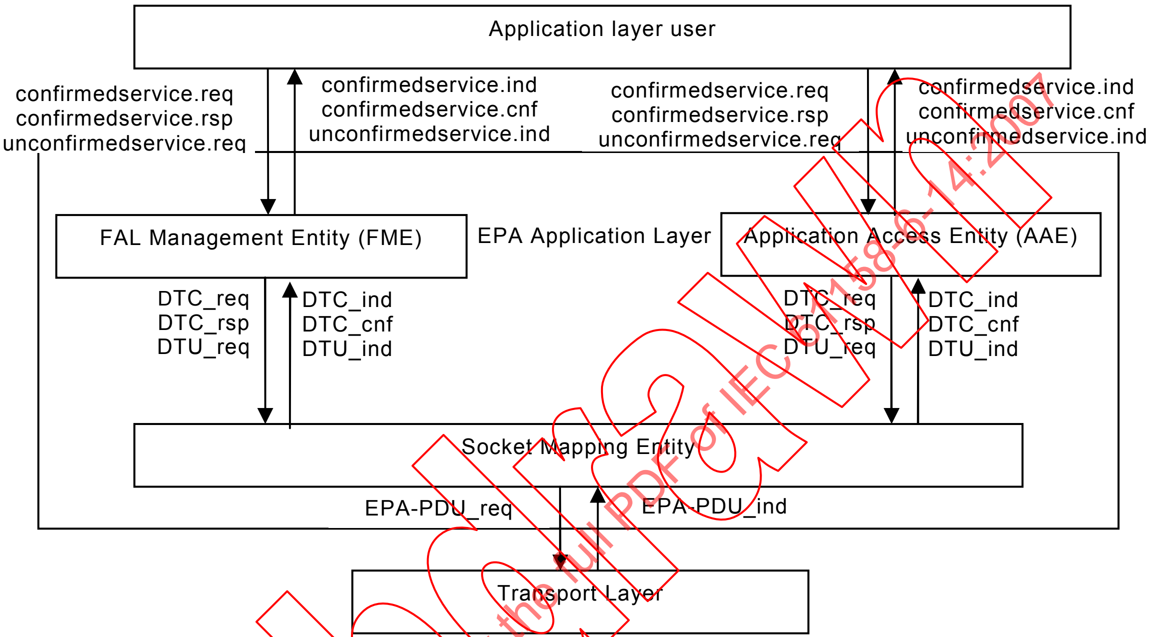

Figure 2 – Exchanged primitives of protocol state machine

7 AP-Context state machine

7.1 Primitives exchanged between ALU and ALE

Table 70 and Table 71 show the primitives exchanged between the Application Layer User (ALU)and Application Layer Entity (ALE).

Table 70 – Primitives delivered by ALU to ALE

Primitive name	Source	Associated parameters	Functions
ConfirmedService.req	ALU	Data, Remote_IP_Address	This primitive is used to send a confirmed service request primitive to ALE by the user of application layer
ConfirmedService.rsp	ALU	Data, Remote_IP_Address	This primitive is used to send a confirmed service response primitive to ALE by the user of application layer
UnconfirmedService.req	ALU	Data, Remote_IP_Address	This primitive is used to send an unconfirmed service request primitive to ALE by the user of application layer

Table 71 – Primitives delivered by ALE to ALU

Primitive name	Source	Associated parameters	Functions
ConfirmedService.ind	ALE	Data, Remote_IP_Address	This primitive is used to send a confirmed service indication primitive to the user of application layer by ALE
ConfirmedService.cnf	ALE	Data, Remote_IP_Address	This primitive is used to send a confirmed service confirmation primitive to the user of application layer by ALE
UnconfirmedService.ind	ALE	Data, Remote_IP_Address	This primitive is used to send an unconfirmed service indication primitive to the user of application layer by ALE

7.2 Protocol state machine descriptions

The AP Context Entity of Type 14 is always active. Its state is described in Table 72.

Table 72 – ACE state descriptions

States	Descriptios
ACTIVE	The ACEs in ACTIVE state prepare to transfer service primitives to ALU and FME(or AAE), or to receive primitives from ALU and FME(or AAE)

7.3 State transitions

The state transitions of ACE are defined in Figure 3, Table 73 and Table 74.

Figure 3 – ACE protocol state machine

Table 73 – ACE state transitions (sender)

#	Current state	Event or condition => action	Next state

S1	ACTIVE	DomainDownload.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S2	ACTIVE	DomainUpload.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S3	ACTIVE	AcknowlodgeEventNotification.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S4	ACTIVE	AlterEventConditionMonitor.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S5	ACTIVE	Read.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S6	ACTIVE	Write.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S7	ACTIVE	EM_GetDeviceAttribute.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S8	ACTIVE	EM_SetDeviceAttribute.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

S9	ACTIVE	EM_ClearDeviceAttribute.req
		=>
		confirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S10	ACTIVE	DomainDownload.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S11	ACTIVE	DomainUpload.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S12	ACTIVE	AcknowlodgeEventNotification.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S13	ACTIVE	AlterEventConditionMonitor.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S14	ACTIVE	Read.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S15	ACTIVE	Write.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S16	ACTIVE	EM_GetDeviceAttribute.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S17	ACTIVE	EM_SetDeviceAttribute.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

S18	ACTIVE	EM_ClearDeviceAttribute.rsp
		=>
		confirmedservice.rsp {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S19	ACTIVE	EventNotification.req
		=>
		unconfirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S20	ACTIVE	Distribute.req
		=>
		unconfirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S21	ACTIVE	EM_FindTagQuery.req
		=>
		unconfirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S22	ACTIVE	EM_FindTagReply.req
		=>
		unconfirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S23	ACTIVE	EM_DeviceAnnunciation.req
		=>
		unconfirmedservice.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

Table 74 – ACE state transitions (receiver)

#	Current state	Event or condition => action	Next state

R1	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “Domain Download”
		=>
		DomainDownload.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R2	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “Domain Upload”
		=>
		DomainUpload.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R3	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “Acknowlodge Event Notification”
		=>
		AcknowlodgeEventNotification.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R4	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “Alter Event Condition Monitor”
		=>
		AlterEventConditionMonitor.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R5	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “Read”
		=>
		Read.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R6	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “Write”
		=>
		Write.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R7	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “EM_GetDeviceAttribute”
		=>
		EM_GetDeviceAttribute.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

R8	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “EM_SetDeviceAttribute”
		=>
		EM_SetDeviceAttribute.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R9	ACTIVE	Confirmedservice.ind
		APServiceType(data) = “EM_ClearDeviceAttribute”
		=>
		EM_ClearDeviceAttribute.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R10	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “Domain Download”
		=>
		DomainDownload.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R11	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “Domain Upload”
		=>
		DomainUpload.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R12	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “Acknowlodge Event Notification”
		=>
		AcknowlodgeEventNotification.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R13	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “Alter Event Condition Monitor”
		=>
		AlterEventConditionMonitor.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R14	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “Read”
		=>
		Read.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

R15	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “Write”
		=>
		Write.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R16	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “EM_GetDeviceAttribute”
		=>
		EM_GetDeviceAttribute.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R17	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “EM_SetDeviceAttribute”
		=>
		EM_SetDeviceAttribute.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R18	ACTIVE	Confirmedservice.cnf
		APServiceType(data) = “EM_ClearDeviceAttribute”
		=>
		EM_ClearDeviceAttribute.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R19	ACTIVE	UnconfirmedService.ind
		APServiceType(data) = “EventNotification”
		=>
		EventNotification.ind {
		Data := user_data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R20	ACTIVE	UnconfirmedService.ind
		APServiceType(data) = “Distribute”
		=>
		Distribute.ind {
		Data := user_data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R21	ACTIVE	UnconfirmedService.ind
		APServiceType(data) = “EM_FindTagQuery”
		=>
		EM_FindTagQuery.ind {
		Data := user_data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

R22	ACTIVE	UnconfirmedService.ind
		APServiceType(data) = “EM_FindTagReply”
		=>
		EM_FindTagReply.ind {
		Data := user_data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R23	ACTIVE	UnconfirmedService.ind
		APServiceType(data) = “EM_DeviceAnnunciation”
		=>
		EM_DeviceAnnunciation.ind {
		Data := user_data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

7.4 Function descriptions

Table 75 describes the functions used by ACE state transitions.

Table 75 – APServiceType() descriptions

Name	APServiceType	Used in	ACE
Input		**Output**	
data		Receive the service type of service message	
Function			
To judge the message type by receiving service message, including domain download, domain upload, acknowledge event notification, alert event condition monitor, read, write, EM_GetDeviceAttribute, EM_SetDeviceAttribute, EM_ClearDeviceAttribute, Event Notification, Distribute, EM_FindTagQuery, EM_FindTagReply and EM_DeviceAnnunciation			

8 FAL management state machines

8.1 Primitives

8.1.1 Primitives exchanged between FME and application layer user

Table 76 and Table 77 show the primitives exchanged between FME and application layer user.

Table 76 – Primitives delivered by application layer user to FME

Primitive name	Source	Associated parameters	Functions
ConfirmedService.req	User of Application Layer	Data, Remote_IP_Address	This primitive is used to send a confirmed service request primitive to FME by the user of application layer
ConfirmedService.rsp	User of Application Layer	Data, Remote_IP_Address	This primitive is used to send a confirmed service response primitive to FME by the user of application layer
UnconfirmedService.req	User of Application Layer	Data, Remote_IP_Address	This primitive is used to send an unconfirmed service request primitive to FME by the user of application layer

Table 77 – Primitives delivered by FME to application layer user

Primitive name	Source	Associated parameters	Functions
ConfirmedService.ind	FME	Data, Remote_IP_Address	This primitive is used to send a confirmed service indication primitive to the user of application layer by FME
ConfirmedService.cnf	FME	Data, Remote_IP_Address	This primitive is used to send a confirmed service confirmation primitive to the user of application layer by FME
UnconfirmedService.ind	FME	Data, Remote_IP_Address	This primitive is used to send an unconfirmed service indication primitive to the user of application layer by FME

8.1.2 Primitive parameters of FME and user of application layer

Table 78 shows the primitives parameters of FME and the user of application layer.

Table 78 – Primitive parameters exchanged between FME and application layer user

Parameter name	Description
remote_ip_address	This parameter transfers the IP address of remote device, namely the destination address sent by sender and the source address received by receiver
Data	This paramenter trasfers the data sent by sender and the data received by receiver

8.1.3 Primitives exchanged between FME and ESME

Table 79 and Table 80 show the primitives exchanged between FME and ESME.

Table 79 – Primitives delivered by FME to ESME

Primitive name	Source	Associated parameters	Functions
DTC_req	FME	remote_ip_address, Data	This primitive is used to send a confirmed service request primitive to ESME by FME
DTC_rsp	FME	remote_ip_address, Data	This primitive is used to send a confirmed service response primitive to ESME by FME.
DTU_req	FME	remote_ip_address, Data	This primitive is used to send a unconfirmed service request primitive to ESME by FME.

Table 80 – Primitives delivered by ESME to FME

Primitive name	Source	Associated parameters	Functions
DTC_ind	EPA Socket Mapping Entity	remote_ip_address, data	This primitive is used to send a confirmed service indication primitive to FME by ESME
DTC_cnf	EPA Socket Mapping Entity	remote_ip_address, data	This primitive is used to send a confirmed service acknowlage primitive to FME by ESME
DTU_ind	EPA Socket Mapping Entity	remote_ip_address, data	This primitive is used to send a unconfirmed service indication primitive to FME by ESME

8.1.4 Primitive parameters exchanged between FME and ESME

Table 81 shows the primitives parameters exchanged between FME and ESME.

Table 81 – Primitives parameters exchanged between FME and ESME

Parameter name	Description
remote_ip_address	This parameter transfers the IP address of remote device, namely the destination address sent by sender and the source address received by receiver
Data	This paramenter transfers the data sent by sender and the data received by receiver

8.2 Protocol state machine descriptions

The states of EPA devices can be one of *No address*, *Unconfigured* or *Configured*. The protocol state machine is shown in Figure 4.

8.2.1 No address

In this state, the EPA device waits for IP address assignment. The IP address can be appointed statically by user or assigned by DHCP server. After the device gets its IP address through DHCP, it will change its next state into *Unconfigured* or *configured* depending on its state when it was powered off before. That is, if the state when the device powered off before is *Configured*, then the next state is *Configured*, vice versa.

8.2.2 Unconfigured

In this state, the FAL Management Entity (FME) uses the specific multicast destination IP address to send EM_ActiveNotification request message to inform other devices of its presence. When this request message is received, user configuration application can query, clear or set the attributes of this device using EM_DetectingDevice, EM_ConfiguringDevice and EM_SetDefaultValue services. After configuration, FME will change its state into *Configured* and start normal operation.

8.2.3 Configured

In this state, the device can participate in normal operations. Users can configure its application information using the services provided by application layer to implement specific predefined control function.

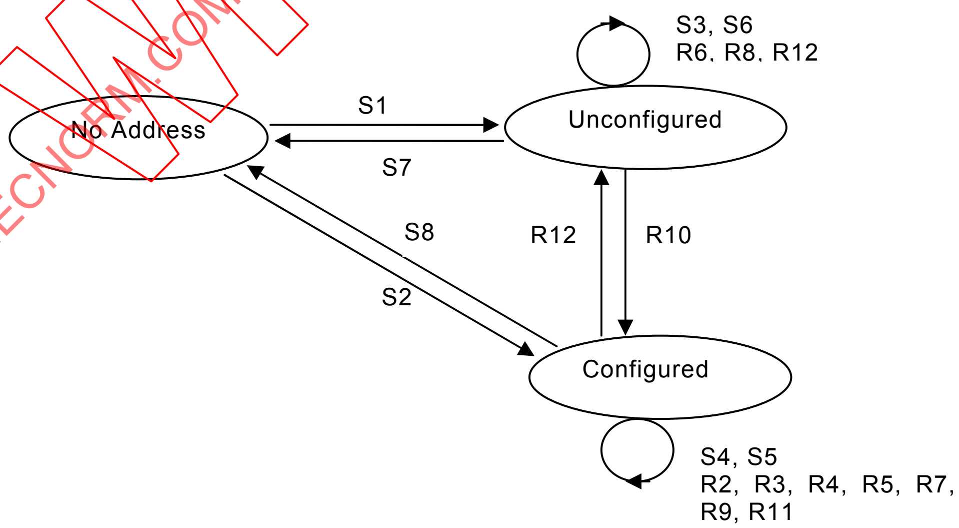

Figure 4 – FME protocol state machine

8.3 State transitions

The state transitions of FME are defined in Figure 4 and Table 82.

Table 82 – State transitions of EPA FME

#	Current state	Event or condition => action	Next state
S1	No Address	EpaRcvNewIpAddress (address) = TRUE && EpaAttribute_Set ()= FALSE => EpaRestoreDefaults () EpaNewAddress(address) EM_ActiveNotification.req {} Restart_EPARepeatTimer()	Unconfigured
S2	No Address	EpaRcvNewIpAddress (address) = TRUE && EpaAttribute_Set()=TRUE => EpaClear_DuplicatePdTagFlag() EpaNewAddress (address) EM_ActiveNotification.req {} Restart_EPARepeatTimer () EM_DetectingDevice.req {}	Configured
S3	Unconfigured	EpaRepeatTimerExpires () => EM_ActiveNotification.req {} Restart_EPARepeatTimer ()	Unconfigured
S4	Configured	EM_UnconfirmedService.req {} || EM_ConfirmedService.req {} || EM_ConfirmedService.rsp {} => EpaSend_EM_ReqRspMessage (em_svc)	Configured
S5	Configured	EM_ConfirmedService.err {} => EpaSend_EM_CommonErrorRsp ()	Configured
S6	Unconfigured	EpaSntpSyncLost () => EM_ActiveNotification.req {} Restart_EPARepeatTimer ()	Unconfigured
S7	Unconfigured	EpaIPAddressCollision () = TRUE = > (no actions taken)	No Address
S8	Configured	EpaIPAddressCollision () = TRUE = > (no actions taken)	No Address
R1	Unconfigured	EpaRecvMsg () ="Any Confirmed EPA_FME Rsp Message" || EpaRecvMsg ()="Any Confirmed EPA_FME Error Message" => EM_ConfirmedService.cnf{}	Unconfigured
R2	Configured	EpaRecvMsg()=" EM_DetectingDevice" && EpaQueryMatch (em_svc) = TRUE => EM_OnlineReply.req {}	Configured
R3	Configured	EpaRecvMsg()=" EM_OnlineReply" && Epa_MessageIdMatch(em_svc) = TRUE && EpaDeviceId_Match (em_svc) = FALSE => EPASet_DuplicatePdTagFlag ()	Configured

#	Current state	Event or condition => action	Next state
		EM_ActiveNotification.req {} Restart_EPARepeatTimer ()	
R4	Configured	EpaRecvMsg()=" EM_OnlineReply" && Epa_MessageIDMatch(em_svc) = TRUE && EpaDeviceId_Match (em_svc) = TRUE => //Do nothing-the response is from this device	Configured
R5	Configured	EpaRecvMsg() = " EM_OnlineReply" => EM_OnlineReply.ind {}	Configured
R6	Unconfigured	EpaRecvMsg() = " EM_GetDeviceAttributeReq" => EM_ConfrimedService.err {}	Unconfigured
R7	Configured	EpaRecvMsg()=" EM_GetDeviceAttributeReq" => EM_GetDeviceAttribute.rsp {}	Configured
R8	Unconfigured	EpaRecvMsg()=" EM_ConfiguringDeviceReq" || EpaRecvMsg()=" EM_SetDefaultValueReq" && EpaDeviceId_Match(em_svc)= FALSE => EM_ConfirmedService.err { }	Unconfigured
R9	Configured	EpaRecvMsg()="EM_ConfiguringDeviceReq" || EpaRecvMsg()=" EM_SetDefaultValueReq" && EpaPdTag_Match(em_svc)= TRUE => EM_ConfirmedService.rsp {}	Configured
R10	Unconfigured	EpaRecvMsg()=" EM_ConfiguringDeviceReq" => EpaSet_Attribute_Data(em_svc) EpaClear_DuplicatePdTagFlag () EM_ConfiguringDevice.rsp {} EM_ActiveNotification.req {} Restart_EPARepeatTimer () EM_DetectingDevice.req {}	Configured
R11	Configured	EpaRecvMsg=" EM_GetDeviceAttributeReq" => EM_GetDeviceAttribute.rsp {}	Configured
R12	Configured	EpaRecvMsg()="EM_SetDefaultValueReq" => EpaResoreDefaults () EM_SetDefaultValue.rsp {} EM_ActiveNotification.req {} Restart_EPARepeatTimer ()	Unconfigured

8.4 Function descriptions

The functions used in FME state transitions are listed in Table 83 through Table 100.

NOTE The em_svc represents the message that comes from user configuration programs.

8.4.1 EpaRcvNewIpAddress()

The EpaRcvNewIpAddress() is illustrated in Table 83.

Table 83 – EpaRcvNewIpAddress() descriptions

Name	EpaRcvNewIpAddress	Using	FME
Input		**Output**	
Address		TRUE or FALSE	
Function			
This function is invoked when an IP address is received. The input parameters identify the interface of the device and IP address received.If this function is invoked correctly, then it returns TRUE, otherwise it returns FALSE.			

8.4.2 EpaAttribute_Set()

The EpaAttribute_Set() is illustrated in Table 84.

Table 84 – EpaAttribute_Set() descriptions

Name	EpaAttribute_Set	Using	FME
Input		**Output**	
None		TRUE or FALSE	
Function			
Returns true if the attribute of the EPA device has been set by the EM_ConfiguringDevice service and is currently still valid. Otherwise, returns false.			

8.4.3 EpaRestoreDefaults()

The EpaRestoreDefaults() is illustrated in Table 85.

Table 85 – EpaRestoreDefaults() descriptions

Name	EpaRestoreDefaults	Using	FME
Input		**Output**	
None		None	
Function			
When this function is invoked, all links are disconnected, and the attributes of EME are set as default value.			

8.4.4 EpaNewAddress()

The EpaNewAddress() is illustrated in Table 86.

Table 86 – EpaNewAddress() descriptions

Name	EpaNewAddress	Using	FME
Input		**Output**	
Address		TRUE or FALSE	
Function			
Returns true if the IP address is updated correctly when EPA device received a new IP address. Otherwise, returns false.			

8.4.5 Restart_EPARepeatTimer()

Restart_EPARepeatTimer() is illustrated in Table 87.

Table 87 – Restart_EPARepeatTimer() descriptions

Name	Restart_EPARepeatTimer	Using	FME
Input		**Output**	
None		None	
Function			
This function is invoked to restore and restart EPARepeatTimer.			

8.4.6 EpaClear_DuplicatePdTagFlag()

The EpaClear_DuplicatePdTagFlag() is illustrated in Table 88.

Table 88 – EpaClear_DuplicatePdTagFlag() descriptions

Name	EpaClear_DuplicatePdTagFlag	Using	FME
Input		**Output**	
None		None	
Function			
This function is invoked to clear duplicate detected state.			

8.4.7 EPARepeatTimerExpire()

The EPARepeatTimerExpire() is illustrated in Table 89.

Table 89 – EPARepeatTimerExpire() descriptions

Name	EPARepeatTimerExpire	Using	FME
Input		**Output**	
None		None	
Function			
This function is invoked when the EPA Repeat Timer expires.			

8.4.8 EpaSend_EM_ReqRspMessage()

The EpaSend_EM_ReqRspMessage() is illustrated in Table 90.

Table 90 – EpaSend_EM_ReqRspMessage() descriptions

Name	EpaSend_EM_ReqRspMessage	Using	FME
Input		**Output**	
Em_svc		None	
Function			
Constructs and sends request or response messages.			

8.4.9 EpaSend_EM_CommonErrorRsp()

The EpaSend_EM_CommonErrorRsp() is illustrated in Table 91.

Table 91 – EpaSend_EM_CommonErrorRsp() descriptions

Name	EpaSend_EM_CommonErrorRsp	Using	FME
Input		**Output**	
service_type, spec_params		None	
Function			
Constructs and sends error response information.			

8.4.10 EpaSntpSyncLost()

The EpaSntpSyncLost() is illustrated in Table 92.

Table 92 – EpaSntpSyncLost() descriptions

Name	EpaSntpSyncLost	Using	FME
Input		**Output**	
None		None	
Function			
This function is invoked when the state of synchronization of time between the device and the selected remote time server changes from synchronization to no –synchronization.			

8.4.11 EpaIPAddressCollision()

The EpaIPAddressCollision() is illustrated in Table 93.

Table 93 – EpaIPAddressCollision() descriptions

Name	EpaIPAddressCollision	Using	FME
Input		**Output**	
None		TRUE or FALSE	
Function			
If the IP address is conflict with others, return TURE. Otherwise, return FALSE.			

8.4.12 EpaRecvMsg()

The EpaRecvMsg() is illustrated in Table 94.

Table 94 – EpaRecvMsg() descriptions

Name	**EpaRecvMsg**	**Using**	**FME**
Input		**Output**	
Em_svc		Em_svc message type	
Function			
This function is invoked when a message is received. It decodes the em_svc and returns the type of the EPA management service.			

8.4.13 EpaQueryMatch()

The EpaQueryMatch() is illustrated in Table 95.

Table 95 – EpaQueryMatch() descriptions

Name	**EpaQueryMatch**	**Using**	**FME**
Input		**Output**	
Em_svc		TRUE or FALSE	
Function			
Returns TRUE if the queried object is contained in the device.			

8.4.14 EpaMessageIDMatch()

The EpaMessageIDMatch() is illustrated in Table 96.

Table 96 – EpaMessageIDMatch() descriptions

Name	**EpaMessageIDMatch**	**Using**	**FME**
Input		**Output**	
Em_svc		TRUE or FALSE	
Function			
Returns TRUE if the MessageID contained in the messages matches that contained in the EM_DetectingDevice service.			

8.4.15 EpaDeviceId_Match()

The EpaDeviceId_Match() is illustrated in Table 97.

Table 97 – EpaDevId_Match() descriptions

Name	EpaDeviceId_Match	Using	FME
Input		**Output**	
em_svc		TRUE or FALSE	
Function			
Returns true if the Device ID in the message exactly matches the Device ID of this device.			

8.4.16 EpaPdTag_Match()

The EpaPdTag_Match() is illustrated in Table 98.

Table 98 – EpaPdTag_Match() descriptions

Name	EpaPdTag_Match	Using	FME
Input		**Output**	
em_svc		TRUE or FALSE	
Function			
Returns TRUE if the PD_Tag contained in the message matches the PD_Tag of the device.			

8.4.17 EpaSet_Attribute_Data()

The EpaSet_Attribute_Data() is illustrated in Table 99.

Table 99 – EpaSet_Attribute_Data() descriptions

Name	EpaSet_Attribute_Data	Using	FME
Input		**Output**	
em_svc		None	
Function			
Updates the attributes of the EME in the EPA device.			

8.4.18 EpaSet_DuplicatePdTagFlag()

The EpaSet_DuplicatePdTagFlag() is illustrated in Table 100.

Table 100 – EpaSet_DuplicatePdTagFlag() descriptions

Name	EpaSet_DuplicatePdTagFlag	Using	FME
Input		**Output**	
None		None	
Function			
The value of DuplicateDetectedState is set if the device has detected that another device has the same PD_Tag.			

9 Application access entity protocol machine

9.1 Primitives

9.1.1 Primitives exchanged between AAE and application layer user

The primitives exchanged between AAE and Application Layer User (ALU) are shown in Table 101 and Table 102.

Table 101 – Primitives issued by ALU to AAE

Primitive name	Source	Associated parameters	Functions
ConfirmedService.req	ALU	remote_ip_address, data	This is a primitive used to convey a ConfirmedService request primitive from the ALU to the AAE
ConfirmedService.rsp	ALU	remote_ip_address, data	This is a primitive used to convey a ConfirmedService response primitive from the ALU to the AAE
UnconfirmedService.req	ALU	remote_ip_address, data	This is a primitive used to convey a UnconfirmedService request primitive from the ALU to the AAE

Table 102 – Primitives issued by AAE to ALU

Primitive name	Source	Associated parameters	Functions
ConfirmedService.ind	AAE	remote_ip_address, data	This is a primitive used to convey a ConfirmedService indication primitive from the AAE to the ALU
ConfirmedService.cnf	AAE	remote_ip_address, data	This is a primitive used to convey a ConfirmedService confirmation primitive from the AAE to the ALU
UnconfirmedService.ind	AAE	remote_ip_address, data	This is a primitive used to convey a UnconfirmedService indication primitive from the AAE to the ALU

9.1.2 Primitive parameters exchanged between AAE and ALU

Table 103 describes the parameters used with primitives exchanged between AAE and ALU.

Table 103 – Primitives parameters exchanged between AAE and ALU

Parameter name	Description
remote_ip_address	This parameter transfer the IP address of the remote device,namely the destination address that the sender will send data to and the source address that the receiver will receive data from
data	This parameter transfer the data that the sender will send and the receiver will receive

9.1.3 Primitives Exchanged between AAE and ESME

The primitives exchanged between AAE and ESME are shown in Table 104 and Table 105.