PUBLICLY
AVAILABLE
SPECIFICATION

IEC PAS 62409

First edition
2005-06

Real-time Ethernet for Plant Automation (EPA®)

Reference number
IEC/PAS 62409:2005(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**

- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/searchpub) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/online_news/ justpub) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

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

PUBLICLY
AVAILABLE
SPECIFICATION

IEC PAS 62409

First edition
2005-06

Real-time Ethernet for Plant Automation (EPA®)

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XJ

For price, see current catalogue

CONTENTS

FIGURES

TABLES

INTRODUCTION

Considering the increasingly use of information technology (IT) with established standards, such as TCP/IP and XML, in modern industrial automation, this PAS provides a EPA® (Ethernet for Plant Automation) system architecture and communication services and protocols specification to meet the demand of deterministic communication based on the commonly known as Ethernet. EPA network uses provision from ISO/IEC 8802-3:2000 for the lower communication stack layers and additionally provide more predictable and reliable real-time data transfer and means for support of precise synchronization of automation equipment.

It contains the following items:

1) EPA system architecture
2) Date Link Layer protocol specification
3) Application Layer Service definition
4) Application Layer protocol specification
5) XML based EPA Device Description specification

In EPA systems, regular Ethernet traffic is supported in parallel. For that purpose of higher transmission priority, a registered number 0X88BC for LENGTH/TYPE segment is used.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

Real-time Ethernet for Plant Automation (EPA®)

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.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent concerning EPA®[1].

The EPA has the patent applications listed below:

China Publication Number 03142040.0

China Publication Number 031200001.9.

IEC takes no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured the IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with IEC. Information may be obtained from:

ZHEJIANG SUPCON CO. LTD.
Liuhe Road,
Binjiang District,
Hangzhou, China.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

---

[1] EPA® is the trade name of ZHEJIANG SUPCON CO. LTD., control of trade name use is given to IEC and the non profit organisation EPA Club. This information is given for the convenience of users of this PAS and does not constitute an endorsement by IEC of the trademark holder or any of its products. Use of the trade name EPA should require permission of the EPA Club.

A PAS is a technical specification not fulfilling the requirements for a standard but made available to the public .

IEC-PAS 62409 has been processed by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

The text of this PAS is based on the following document:	This PAS was approved for publication by the P-members of the committee concerned as indicated in the following document
Draft PAS	**Report on voting**
65C/357/NP	65C/373/RVN

Following publication of this PAS, the technical committee or subcommittee concerned will transform it into an International Standard.

It is intended that the content of this PAS will be incorporated in the futures new editions of the various parts of IEC 61158 series and/or IEC 61784 series according to the structure of these series.

This PAS shall remain valid for an initial maximum period of three years starting from 2005-06. The validity may be extended for a single three-year period, following which it shall be revised to become another type of normative document or shall be withdrawn.

Real-time Ethernet for Plant Automation (EPA®)

1 Scope

This PAS defines the EPA (Ethernet for Plant Automation) system structure, data link layer protocol, application layer service definition and protocol based on ISO/IEC 8802-3:2000, RFC 791, RFC 768 and so on. XML-based device description is also defined.

This PAS can be used for manufacturing and process automation.

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 61588:2004, *Precision clock synchronization protocol for networked measurement and control system*

IEC 61158 (all parts), *Digital data communications for measurement and control – Fieldbus for use in industrial control systems*

IEC 61158-5:2003, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 5: Application layer service definition*

IEC 61784-1:2003, *Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems*

IEC 61499 (all parts), *Function blocks*

IEC 61499-1 *Function Blocks for industrial-process measurement and control systems – Part 1 – Architecture*

IEC 61804 (all parts), *Function blocks (FB) for process control*

IEC 61804-1, *Function Blocks for process control – Part 1: Overview of system aspects*

IEC 61804-2, *Function Blocks for process control – Part 2: Specification of FB concept and Electronic Device Description Language (EDDL)*

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

ISO/IEC 8802-3:2000, *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 8824-1:2002 *Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation*

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

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

ISO 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*

ISO 2375:2003, *Information technology – Procedure for registration of escape sequences and coded character sets*

IEEE Std 754:1985, *Binary floating-point arithmetic*

RFC 768, *User Datagram Protocol*

RFC 791, *Internet protocol*

RFC 792, *Internet Control Message Protocol*

RFC 793, *Transmission Control Protocol*

RFC 826, *An Ethernet Address Resolution Protocol*

RFC 919, *Broadcasting Internet Datagrams*

RFC 922, *Broadcasting Internet Datagrams In the Presence of Subnets*

RFC 959, *File Transfer Protocol (FTP)*

RFC 1112, *Host Extensions for IP Multicasting*

RFC 1157, *A Simple Network Management Protocol (SNMP)*

RFC 1533, *DHCP Options and BOOTP Vendor Extensions*

RFC 1541, *Dynamic Host Configuration Protocol (DHCP)*

RFC 2030, *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*

3 Common terms and definitions

3.1 ISO/IEC 7498-1 Terms

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

3.1.1 application entity

3.1.2 application protocol data unit

3.1.3 application service element

3.1.4 communication stack

3.1.5 function block

3.1.6 function block instance

3.1.7 functional unit

3.1.8 interoperability

3.1.9 publisher/subscriber

3.1.10 client/server

3.1.11 report distribution

3.1.12 socket

3.1.13 state machine

3.1.14 transfer syntax

3.2 ISO/IEC 8824-1 Terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824-1 apply.

3.2.1 object Identifier

3.2.2 Type

3.2.3 DL-Scheduling-policy

3.2.4 DLCEP

3.2.5 DLPDU

3.2.6 DLSDU

3.2.7 DLSAP

3.2.8 link

3.2.9 network address

3.2.10 node address

3.2.11 tag

3.2.12 scheduled

3.2.13 unscheduled

3.2.14 frame

3.3 ISO/IEC 10731 Terms

For the purposes of this document, the following terms as defined in ISO/IEC 10731 apply.

3.3.1 request primitive

3.3.2 indication primitive

3.3.3 response primitive

3.3.4 confirm primitive

3.4 IEC 61158-5 Terms

For the purposes of this document, the following terms as defined in IEC 61158-5 apply.

3.4.1 application

3.4.2 application layer interoperability

3.4.3 application objects

3.4.4 application process

3.4.5 application process identifier

3.4.6 application process object

3.4.7 application process object class

3.4.8 application relationship

3.4.9 application relationship application service element

3.4.10 application relationship endpoint

3.4.11 attribute

3.4.12 behaviour

3.4.13 channel

3.4.14 class

3.4.15 class code

3.4.16 class specific service

3.4.17 client

3.4.18 communication objects

3.4.19 connection

3.4.20 connection path

3.4.21 cyclic

3.4.22 data consistency

3.4.23 device

3.4.24 device profile

3.4.25 end node

3.4.26 endpoint

3.4.27 error

3.4.28 error class

3.4.29 error code

3.4.30 event

3.4.31 frame

3.4.32 index

3.4.33 instance

3.4.34 instance attributes

3.4.35 instantiated

3.4.36 network

3.4.37 object

3.4.38 object specific service

3.4.39 peer

3.4.40 physical device

3.4.41 publisher

3.4.42 push publisher

3.4.43 push subscriber

3.4.44 server

3.4.45 service

3.4.46 subscriber

3.5 IEC 61804-2 Terms

For the purposes of this document, the following terms as defined in IEC 61804-2 apply.

3.5.1 Electronic Device Description Compiler

3.5.2 Electronic Device Description (EDD)

3.5.3 Electronic Device Description Interpreter (EDDI)

3.5.4 Electronic Device Description Language (EDDL)

3.6 ISO/IEC 8802-3 Terms

For the purposes of this document, the following terms as defined in ISO/IEC 8802-3 apply.

3.6.1 destination address

3.6.2 frame check sequence

3.6.3 length/type

3.6.4 MAC Frame

3.6.5 pad

3.6.6 source address

3.7 Additional terms and definitions

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

3.7.1
access control
control on the reading and writing of an object.

3.7.2
access Path
the association of a symbolic name with a variable for the purpose of open communication.

3.7.3
communication macrocycle
set of basic cycles needed for a configured communication activity in a macro network segment.

3.7.4
communication scheduling
the algorithms and the operation for data transfers occurring in a deterministic and repeatable manner.

3.7.5
configuration (of a system or device)
step in system design: selecting functional units, assigning their locations and defining their interconnections.

3.7.6
cyclic
repetitive in a regular manner

3.7.7
destination FB Instance
FB instance that receives the specified parameters.

3.7.8
domain
a part of memory used to store code or data.

3.7.9
domain download
a operation to write data in a domain.

3.7.10
domain upload
a operation to read data from a domain.

3.7.11
entity
particular thing, such as a person, place, process, object, concept, association, or event.

3.7.12
EPA 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.

3.7.13
identifier
16-bit word associated with a system variable.

3.7.14
index
address of an object within an application process.

3.7.15
instance
the actual physical occurrence of an object within a class that identifies one of many objects within the same object class.

3.7.16
instantiation
creation of an instance of a specified type.

3.7.17
management information
network-visible information for the purpose of managing the field system.

3.7.18
management information Base
organized list of the management information.

3.7.19
mapping
set of values having defined correspondence with the quantities or values of another set.

3.7.20
member
piece of an attribute that is structured as an element of an array

3.7.21
message filtering
decision on a message according to a special rule.

3.7.22
micro segment
part of a network, where special scheduling is implemented.

3.7.23
offset
number of octets from a specially designated position.

3.7.24
phase
elapsed fraction of a cycle, measured from some fixed origin

3.7.25
process interface
the data exchange and information mapping between physical process and application unit.

3.7.26
real-time
the ability of a system to provide a required result in a bounded time.

3.7.27
real-time communication
transfer of data in real-time.

3.7.28
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.

3.7.29
schedule
temporal arrangement of a number of related operations.

3.7.30
scheduling macrocycle
time interval to implement a specific schedule.

3.7.31
source FB Instance
the FB instance that sends a specific parameter.

3.7.32
time offset
time difference from a specially designated time.

4 Abbreviated terms and acronyms

AAE	Application Access Entity
AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
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	EPA communication scheduling management entity
EPA	Ethernet for Plant Automation
EM_	(as a prefix) EPA Management
ESME	EPA Socket Mapping Entity
FB	Function Block
FBAP	Function Block Application Process
Ind	Indication
IP	Internet Protocol
LLC	Logical Link Control
LMP	Link Management Protocol
MAC	Medium Access Control
MAU	Medium Attachment Unit
MIB	Management Information 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
XDDL	Extensible Device Description
XDDL	Extensible Device Description Language
XML	Extensible Markup Language
.cnf	Confirm Primitive
.ind	Indication Primitive
.req	Request Primitive
.rsp	Response Primitive

5 Conventions

5.1 Convention for Object definitions

Each object defined in this standard is an instance of its corresponding class. The attributes of a class indicate the characteristics and properties of its corresponding objects. The following template is used for class definition:

EPA ASE: **ASE NAME**
CLASS: **CLASS NAME**
CLASS ID: **CLASS ID**
PARENT CLASS: **PARENT CLASS NAME**
ATTRIBUTES:

1.	(m)	key attribute:	attribute name 1
2.	(m)	attribute:	attribute name 2
3.	(o)	attribute:	attribute name 3

SERVICES:

1.	(o)	Opsservice:	service name 1
2.	(o)	Opsservice:	service name 2

(1) The " EPA ASE:" entry is the name of the EPA ASE that provides the services for the class being specified.

(2) The "CLASS:" entry is the name of the class being specified. All objects defined using this template will be an instance of this class. The class may be specified by this standard, or by a user of this standard.

(3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the EPA ASE that will provide the services for this class. When qualified by the identity of its EPA ASE, it unambiguously identifies the class within the scope of the EPA AL. The value "NULL" indicates that the class cannot be instantiated.

(4) The "PARENT CLASS:" entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited by the class being defined, and therefore do not have to be redefined in the template for the class.

NOTE The parent-class "TOP" indicates that the class being defined is an initial class definition. The parent class TOP is used as a starting point from which all other classes are defined. The use of TOP is reserved for classes defined by this standard.

(5) The "ATTRIBUTES" label indicates that the following entries are attributes defined for the class.

a) Each of the attribute entries contains a line number in column 1, a mandatory (m) /optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label in column 3, a name or a conditional expression in column 4.

b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.

(6) The "SERVICES" label indicates that the following entries are services to be used for the class. An (m) in column 2 indicates that the service is mandatory for the class, while an (o) indicates that it is optional. A (c) in this column indicates that the service is conditional. When all services defined for a class are defined as optional, at least one has to be selected when an instance of the class is defined.

5.2 Conventions for services definitions

5.2.1 General

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

5.2.2 Service parameters

Service primitives are used to represent service user/service provider interactions (ISO/IEC 10731). They convey parameters which indicate information available in the user/provider interaction. In any particular interface, not all parameters need be explicitly stated. The service specifications of this specification use a tabular format to describe the component parameters of the ASE service primitives (see Table 1).

Table 1 – service primitive format

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
Parameter	M	M(=)		
Parameter	M	M(=)		
Parameter	M	M(=)		
Parameter	U	U(=)		
Result(+)			S	S(=)
Parameter			M	M(=)
Result(-)			S	S(=)
Parameter			M	M(=)
Error Type			M	M(=)

The parameters which apply to each group of service primitives are set out in tables. Each table consists of up to five columns for the

1) parameter name,
2) request primitive (.req),
3) indication primitive(.ind)
4) response primitive(.rsp), and
5) confirm primitive(.cnf).

One parameter (or component of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the column:

M parameter is mandatory for the primitive

U parameter is a User option, and may or may not be provided depending on dynamic usage of the service user. When not provided, a default value for the parameter is assumed.

C parameter is conditional upon other parameters or upon the environment of the service user.

— (blank) parameter is never present.

S parameter is a selected item.

Some entries are further qualified by items in parentheses. These may be

a) a parameter-specific constraint:

"(=)" indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.

b) an indication that some note applies to the entry:

"(n)" indicates that the following note "n" contains additional information pertaining to the parameter and its use.

5.3 Conventions for state machines

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.

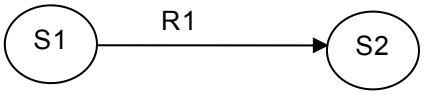


Figure 1 – state transition diagram

Table 2 – 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 2) 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.

6 EPA system architecture

6.1 Overview

The EPA system is a distributed system which uses the Ethernet network defined by ISO/IEC8802-3 to connect field devices and small systems, and to control/monitor equipment in the industrial field. EPA devices work together to provide I/O and control for automated processes and operations.

The EPA system architecture provides a framework for describing these systems as the collection of physical devices interconnected by an EPA network. The objective of this clause is to identify the components of the system, describe their relationships and interactions, and define how they are configured.

EPA networks may be composed of one or more of micro-segments which use standard Ethernet/IP/UDP.

6.2 EPA architecture

6.2.1 Relationship to the ISO OSI Reference Model

Based on the concepts developed in ISO/IEC 7498-1, EPA uses Ethernet/IP/UDP protocols as the lower four communication stack layers, and also defines the interface using time-sharing-controlled transmission method between data link layer defined in IEC/ISO 8802-3 and network layer defined in RFC 791 – internet protocol – for the purpose of deterministic communication. Besides, EPA application layer services and protocols are defined so that interoperation between devices can be implemented.

The relationship between ISO OSI reference model and EPA communication hierarchy is illustrated in Table 3.

Table 3 – Relationship to the ISO/OSI reference model

ISO layers	EPA layers
	EPA User Layer
Application layer	EPA Application layer
Presentation layer	
Session layer	
Transport layer	TCP/UDP
Network layer	IP
Data link layer	EPA communication scheduling management entity ISO/IEC 8802-3
Physical layer	ISO/IEC 8802-3

6.2.1.1 Physical and data link layer

The EPA physical and data link layers use ISO/IEC 8802-3 protocols.

In addition, EPA defines EPA Communication Scheduling Management Entity (ECSME) to manage the deterministic communication, (see clause 7).

6.2.1.2 Network layer and transport layer

The following protocols are used in EPA network layer and transport layer:

a) RFC 791, Internet protocol (IP).
b) RFC 826, Address Resolution Protocol (ARP).
c) RFC 792, Internet Control Message Protocol (ICMP).
d) RFC 1112, Internet Group Manage Protocol (IGMP).
e) RFC 768, User Datagram Protocol(UDP).
f) RFC 793, Transport Control Protocol (TCP).

6.2.1.3 Application layer

The following protocols are used in EPA application layer:

a) RFC 1157, simple network management protocol(SNMP);
b) RFC 959, file transfer protocol(FTP);
c) RFC 1541, dynamic host configure protocol (DHCP);
d) RFC 1533, DHCP options and BOOTP manufacturer extended protocols.
e) RFC 2030, simple network time protocol(SNTP);
f) IEC 61588, Precision Clock-Synchronization Protocol for Networked Measurement and Control Systems (IEEE Standard).

6.2.1.4 User layer

EPA user layer uses *FUN*CTION BLOCKS FOR INDUSTRIAL-PROCESS MEASUREMENT AND CONTROL SYSTEMS defined by IEC 61499 and *FUNCTION BLOCKS FOR PROCESS CONTROL* defined by IEC61804.

EPA user layer also supports the application based on HTTP, FTP, DHCP, SNTP, SNMP etc. which is not included in this specification.

6.2.2 EPA system architecture

6.2.2.1 Overview

The generic components of EPA architecture are illustrated in Figure 2. In EPA systems, the existent protocols, such as ISO/IEC 8802-3, TCP(UDP)/IP, SNMP, SNTP, DHCP, HTTP, FTP, are supported. Additionally, the following entities are defined:

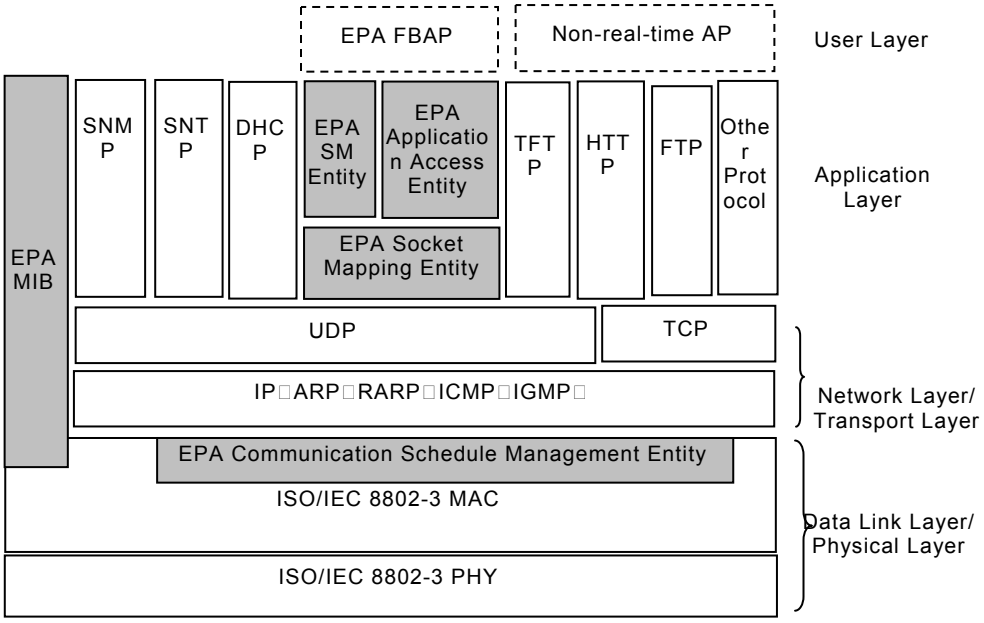


Figure 2 – EPA system architecture

a) Application process,
b) EPA system management entity,
c) EPA application access entity,
d) EPA communication scheduling management entity,
e) EPA management information base, and
f) EPA Socket Mapping Entity.

6.2.2.2 Application process

6.2.2.2.1 Concept

Referring to the ISO OSI Reference Model, the application process is used to describe the portion of a distributed application that is resident in a single device. The term has been adapted to the EPA systems to describe entities within devices that perform a related set of functions that process input/output parameters and events between specific applications in the network.

In EPA systems, there are two kinds of application processes, EPA function block application processes and non-real-time application processes, which may run in parallel in one EPA system.

6.2.2.2.2 Non-real-time application process

Non-real-time application processes are those based on regular Ethernet, such as WEB server, mail application etc..

6.2.2.2.3 EPA function block application process (FBAP)

EPA function block application processes (FBAP) are those based on *FUNCTION BLOCKS FOR INDUSTRIAL-PROCESS MEASUREMENT AND CONTROL SYSTEMS* defined in IEC 61499 and *FUNCTION BLOCKS FOR PROCESS CONTROL* defined in IEC 61804.

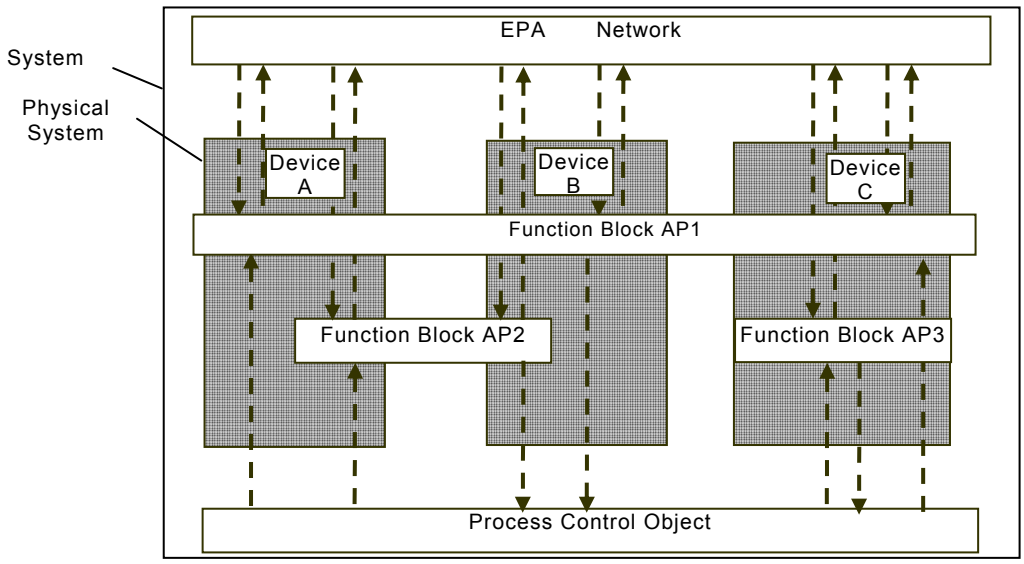
The interoperation between two function blocks is modeled as connecting the input/output parameters between two function blocks using EPA application services.

An EPA application process can be implemented in two ways as shown in Figure 3:

a) All function blocks of one AP are resident in a single device, as the Function Block AP3.

b) Function blocks composing one AP are resident in different devices, as the Function Block AP1 and Function Block AP2 in Figure 3.

Whether to distribute the FBs composing a FBAP into one or more devices is determined according to their physical capabilities and their implementation.



NOTE The process control object is not a part of the EPA system

Figure 3 – EPA control system model

6.2.2.3 EPA application access entity (AAE)

EPA application access entity (AAE) describes the relationship between communication objects, as well as services provided for application layer users and the context transaction interfaces.

AAE provides communication services for FB instances. These services include upload/download, viariable access, events management services, which are used to transfer the measurement and control values, events notification and upload/download programs, etc.

6.2.2.4 EPA system manage entity (SME)

EPA system management entity (SME) manages the communication of EPA devices and integrates these devices into a cooperating communication system.

SME supports the functions of device annunciation, device identification, device location, address assignment, time synchronization, EPA link object management etc. For implementing these functions, SME defines some class objects and services.

6.2.2.5 EPA socket mapping entity (ESME)

EPA socket mapping entity (ESME) conveys SME and AAE services over UDP/IP, provides functions of message transmitting priority management, response maintenance and link status monitoring etc.

6.2.2.6 UDP/IP

UDP/IP protocols are used to transfer the primitives of the SME and AAE services.

6.2.2.7 EPA communication scheduling management entity (ECSME)

EPA communication schedule manage entity (ECSME) provides a time-sharing controlling mechanism to control network access to avoid collision.

6.2.2.8 EPA management information base (MIB)

EPA management information base (MIB) defines the network accessible objects used by SME, ECSME and AAE. For example, device descriptor object describes some information of device ID, Active IP Address etc. Link object describes the access path between two FBs.

6.3 Network Topology

6.3.1 Overview

The EPA network topology is illustrated in Figure 4, including two subnets, process monitor layer (L2) subnet and field device layer (L1) subnet.

The following describes the characteristics of EPA topology:

- L1 subnet is used to connect field devices (such as transmitters, actuators and analytical instruments, etc.) mounted in field environment;
- L1 subnet can be separated into more than one micro-segments, where the time-sharing controlling mechanism defined in clause 7.3 is used to meet the demands of deterministic communication;
- L2 subnet is used to connect the devices of control center and HMI devices and one or more micro-segments. In L2 subnet, regular communication schedule based on CSMA/CD defined in ISO/IEC 8802-3 is applied when deterministic communication is not necessary;
- These devices on both L1 and L2 subnets may be interconnected with standard switches or hubs;

— An EPA device may function as a bridge, which interconnect L1 micro-segment to L2 subnet. This EPA bridge perform message filtering and forwarding between L1 and L2 subnet.

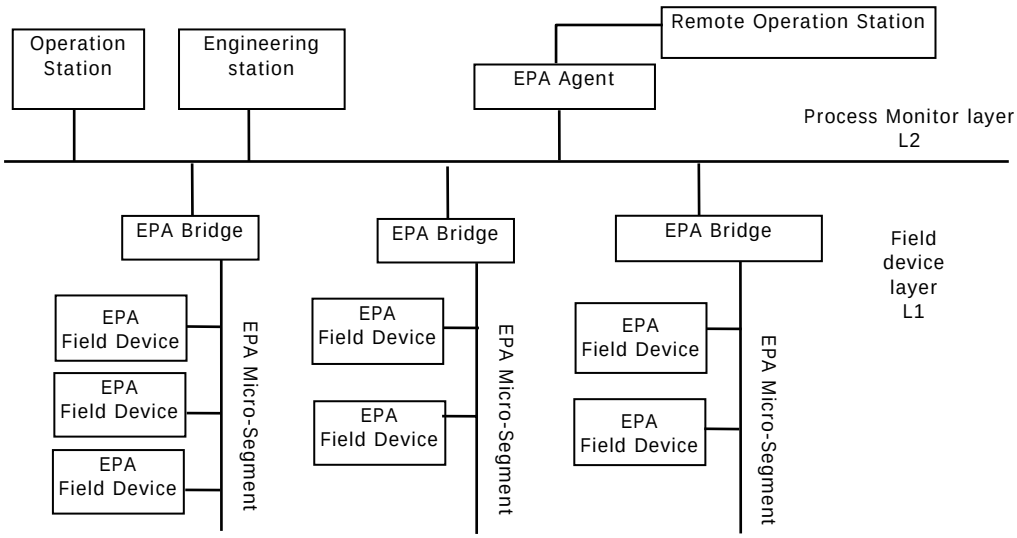


Figure 4 – EPA system network topology

6.3.1.1 Micro-segment

A micro-segment is a control area, where EPA devices communicate with each other to implement specific measuring and control function. That is, the devices which need to communicate with each other shall be interconnected in the same L1 micro-segment.

Example:

Those devices consisting of a control loop or function block application process, such as transmitters, actuators and controllers which need to communicate with each other, must be interconnected in the same micro-segment.

6.3.1.2 Features

An EPA micro-segment has the following features.

topology:star or linear topology;

transmission medium:STP, coaxial line, thick coaxial line, thin coaxial line FDDI;

data transmission speed:10Mbps, 100 Mbps, 1000 Mbps or more high;

transmission distance:

<=500m for thick coaxial wire;

<=185m for thin coaxial wirer

<=100m for STP and UTP;

<=2000m for single model fiber;

<=412m formultimode fiber.

node number: not specified. It is determined by practical applications according to the real-time communication requirement.

6.3.2 EPA devices

6.3.2.1 EPA master device

An EPA master device is that connected to L2 subnet directly. EPA master device has EPA communication interface but may have no control function block or function block application. An EPA master device may commonly be configuration device, monitoring device or HMI station.

An EPA master device has unique IP address in a system.

6.3.2.2 EPA field device

An EPA field Device is that installed in the industrial field environment. EPA field Devices must have EPA communication entity and at least include a function block instance.

An EPA field Device has unique IP address in a system.

6.3.2.3 EPA bridge

An EPA bridge is an optional devices that interconnects one L1 micro-segment to L2 subnet. An EPA bridge at least has two communication interfaces, connecting one L1 micro-segment and L2 subnet respectively.

An EPA bridge can be configured to provide the following functions:

a) Communication isolation

When the traffic occurs between two devices connected in one micro-segment, the EPA bridge shall provide limit it within the segment. Here, the traffic includes broadcast, multicast and peer to peer communication flows.

b) Message forwarding and filtration

An EPA bridge transmit shall forward and filter the messages between one L1 micro-segment and another segment or L2 subnet. That is, when forwarding a message, the EPA bridge shall examine whether it should be forwarded according to a configured criterion.

As an optional device, an EPA bridge is not necessary if the number of nodes in a system is not too large.

6.3.2.4 EPA agent

EPA agent is also an optional device used to interconnect EPA network and other different network. It shall provide the function of security control for the remote access.

6.4 Communication process between EPA devices

6.4.1 Overview

Figure 5 illustrates the communication relationship between two EPA field devices. For the purpose of interoperation, each EPA field device shall include at least one function block instance, EPA application access entity, EPA system management entity, EPA socket mapping entity, EPA link object, communication scheduling management entity and UDP/IP protocol.

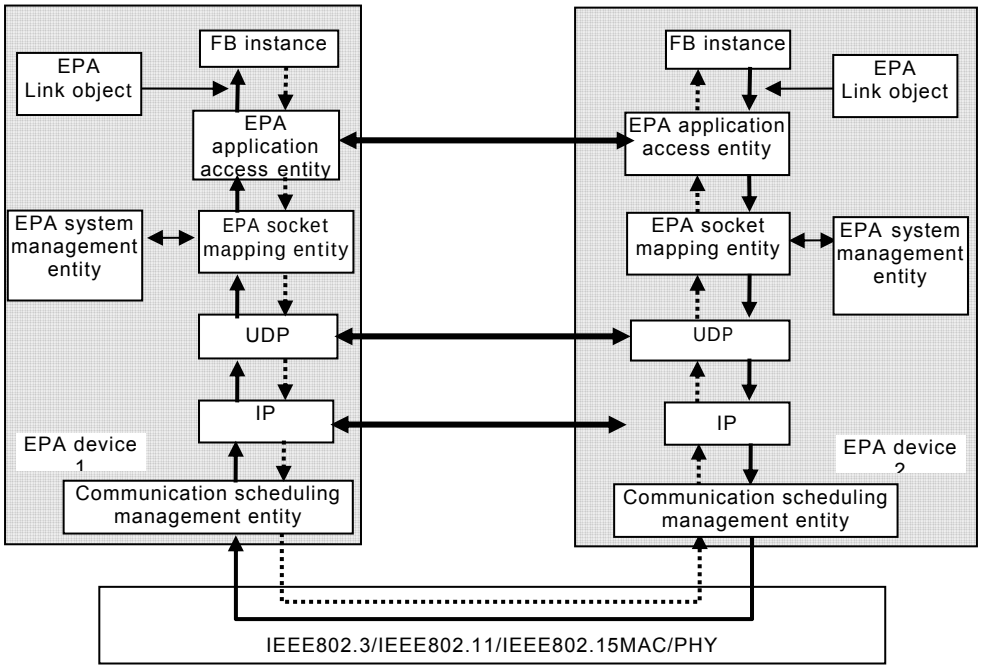


Figure 5 – communication among EPA devices

6.4.2 EPA link object

EPA link object describes the linking relationship or access path between one FB instance's output parameter and another FB instance's input parameter and their communication roles in communication relationship. Each EPA link object is uniquely identified in devices by object ID.

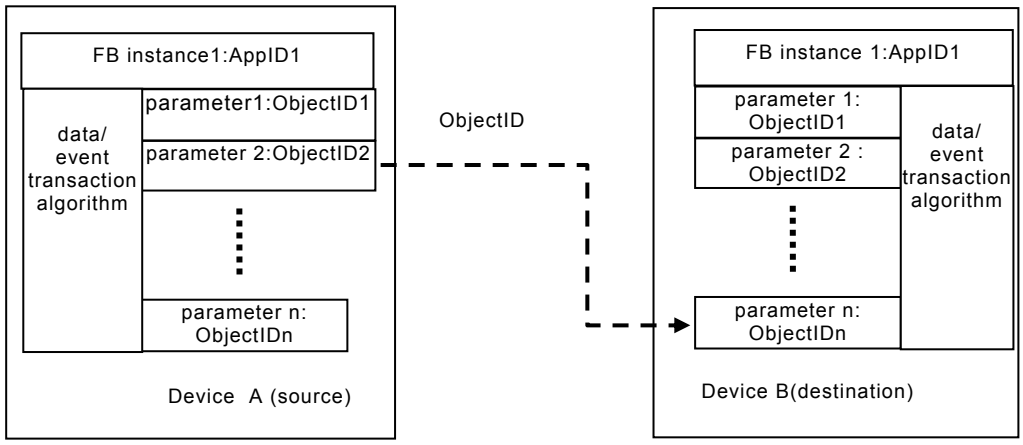


Figure 6 – EPA link relationship

In EPA system, the communication link relationship between the FB instances' input and output parameters is identified by EPA link object with three-level parameters, IP address, application FB instance identifier (AppID) and object index (ObjectID) in the FB instance. That is, an EPA link object describes the source object index with SourceObjectID, source FB instance with SourceAppID, source device with SourceIPAddress as well as destination object index with DestinationObjectID, destination FB instance with DestinationAppID, desination device with DestinationIPAddress in an communication relationship.

EPA link object identifies uniquely the input/output access path between two FB instances (see Figure 6).

6.4.3 An example

The following example illustrates the communication process between the two EPA devices.

Before initiating a communication, each EPA device shall be initialized and establish the connectionless communication relationship according to the downloaded link object configurations.

EPA communication initiator shall handle the communication process as follows:

a) EPA link relationship information inquiring:

EPA initiate side first find the corresponding EPA link object regarding to the ObjectID of the sending parameter, and then inquire the needed information in communication, which include:

1) SourceIPAddress;
2) SourceAppID;
3) SourceObjectID;
4) DestinationIPAddress;
5) DestinationAppID;
6) DestinationObjectID;
7) ServiceID identifying an EPA service to be used;
8) Role(local communication role);

b) Encoding:

According to ServiceID, EPA application process invokes the corresponding EPA application layer service encode user data into EPA service request primitive. Meanwhile, the invoke counter MessageID will increase by 1.

c) Primitive encapsulating

EPA socket mapping entity encapsulates the service primitive by invoking UDP/IP protocol and put it into the buffer.

d) Transmission scheduling

EPA communication scheduling management entity sends the EPA message to the network according to the scheduling criterion defined in clause 7.3.

e) Response maintenance:

For the EPA confirm service request primitive, a response maintenance timer should be started up since the message is sent to the EPA network. For the purpose of several responses maintenance, a response primitive maintenance list should be established and maintained as follows according to the MessageID:

1) If the positive (+) response primitive responding to MessageID is received before the timer runs over, it should be returned to EPA application process with the confirm primitive;

2) If the negative (-) response primitive responding to MessageID is received before the timer runs over, it should be returned to EPA application process with

responding error code. The user should determine whether and when to retransfer the request primitive responding to MessageID;

3) If the timer runs over, a negative response with overtime error code should be returned to application process.

f) Response processing:

EPA application process should process the received response as follows:

1) If receiving a negative response, the user should determine whether and when to retransfer;

2) If positive response is received, it shall be processed according to configuration.

EPA communication receiver shall handle the communication process as follows:

a) waiting to receive:

After the EPA communication controller receives a packet from the network, the EPA communication scheduling management entity shall transfer it directly to the EPA socket mapping entity where the message will be buffered then;

b) message filtrating:

EPA socket mapping entity filtrate the EPA message according to ServiceID encapsulated in the message:

1) If the ServiceID indicates that the received message is a request primitive from remote EPA system management entity, it shall be transferred to local EPA system management entity;

2) If the ServiceID indicates that the received message is a request primitive from remote EPA application access entity, it shall be transferred to local EPA application access entity;

3) if the received message is not that above, it shall be discarded without doing nothing else.

c) EPA message transacting:

Local application process will handle the received primitive as follows:

1) If receiving an unconfirmed request primitive, the EPA application process shall update local parameters using the data from the request primitive.

2) If receiving a confirmed request primitive, the EPA application process shall copy the source link information into destination information. That is: DestinationIPAddress is copied into SourceIPAddress, DestinationAppID into SourceAppID, DestinationObjectID into SourceObjectID. And then, the local source information and the corresponding responding data should be encoded as response primitive and sent to the EPA communication initiator.

6.5 EPA system configuration and startup

6.5.1 Configuration

After an EPA device starts up and enters the operational mode, it can be configured using configuration application. It contains manufacturer configuration, basic information configuration, distributed application configuration and the device configuration.

6.5.1.1 Manufacturer configuration

Manufacturer configuration includes device configuration, EPA protocol version configuration, in-contained FB initialization configuration which shall be solidified in an EPA device when it is manufactured.

6.5.1.2 Basic information configuration

Basic information configuration can be done online or offline according the device capability. It includes the following configurations:

a) IP address and PD_tag configuration

IP address can be dynamically assigned or statically set using configuration program according to the device capability. While IP address is set statically, it shall be guaranteed that the IP collision should be avoided.

Generally, once IP address is assigned, it shall be written into the nonvolatile memory as permanent information until reconfiguration.

b) Control policy configuration

Control policy means that the user shall configure the connection relationship between FBs provided in different devices according to the user's application requirement.

c) Redundancy configuration

If two or more redundant devices are used, it shall be configured which is active and which is backup device as well as their redundancy communication port.

d) DHCP server appointment

As optionally, if the IP address of an EPA device shall be dynamically assigned, the DHCP server should be appointed.

e) Time server appointment

For the purpose of time synchronization, a time server shall be appointed.

In general, a redundant timer server shall be assigned reliability.

6.5.1.3 Distributed application configuration

EPA distributed application configuration shall be done as follows:

a) Communication relationship and scheduling configuration for FBs

According to practical application, the communication relationship between input/output parameters of different FBs shall be configured to form controlling loops. Meanwhile, the

communication scheduling policy shall be configured to determine when updating data will be sent as well.

b) EPA link object configuration

According to the above communication relationship configuration, the EPA link objects shall be written and downloaded into devices.

6.5.2 Device Startup

When an EPA device is powered on, it starts up as follows:

a) A dynamic IP address should be assigned by the DHCP server if the device has no IP address;

b) The EPA device repeatedly will broadcast an unconfirmed device annunciation declaration request message in system through EM_DeviceAnnunciation service. In default case, the EPA device repeats it every 15 seconds until receiving the responding primitive.

c) After User configuration application receives the annunciation primitive, it shall configure the device as follows:

   1) If the PD_Tag of the EPA device is null, the user configuration application should call the EM_SetDeviceAttribute service to set PD_Tag as well as annunciation cycle, redundancy number, redundancy state, redundancy port;

   2) If the PD_Tag of the EPA device isn't null, the user configuration application shall call EM_ClearDeviceAttribute service to clear the device attributes before resetting them.

d) EPA device update local device attributes using the receiving configuration data.

e) Status of the EPA device is changed into operational mode.

7 EPA data link layer protocol

7.1 Overview

This specification adopts data link protocol defined in ISO/IEC 8802-3. In addition, EPA Communication Scheduling Management Entity (ECSME) is defined on ISO/IEC 8802-3 data link protocol to manage the deterministic communication.

ECSME provides the following functions:

a) Transparent data transferring between DLE and DLS_User specified in ISO/IEC 8802-3 without modifying the data;

b) Receiving DLS_User DATA from DLS_User and buffering them;

c) Transferring DLS_User DATA to DLE in configured order and priority. The DLE will send it to Ethernet network using the protocols defined in ISO/IEC8802-3;

d) Transferring decoded DLPDU from DLE to DLS_User.

ECSME supports two ways of communication scheduling:

a) Free competitive communication scheduling based on the CSMA/CD;

b) Deterministic communication based on the time-sharing scheduling policy defined later.

When the former scheduling is used, ECSME shall directly transfer the data between DLE and DLS_User without any buffering or handling.

When the latter b) is used, the ECSME in each EPA device shall transfer DLS_User DATA to DLE according to the pre-configured timing order and priority, the DLE shall process the data and send it to PhL, so that the collision is avoided.

7.2 DLL model

The DLL model is shown in Figure 7.

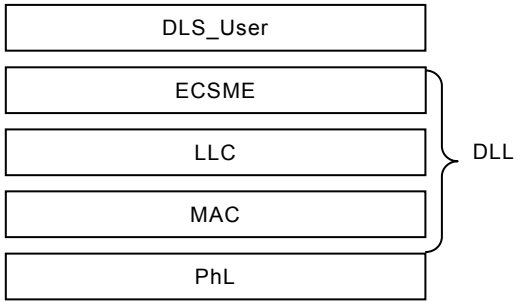


Figure 7 – Model of DLL

7.2.1 Medium Access Control (MAC)

The Medium Access Control (MAC) layer protocol defined in ISO/IEC 8802-3 is applied.

7.2.2 Logic Link Control (LLC)

The Logic Link Control (LLC) protocol defined in ISO/IEC 8802-3 is applied.

7.2.3 EPA Communication Scheduling Management Entity (ECSME)

ECSME is the extension based on LLC defined in ISO/IEC 8802-3. It transfers data between DLS_User and LLC without any changing.

ECSME doesn't alter the services provided by DLL to DLS_User defined in ISO/IEC 8802-3 as well as the interface between PhL and MAC. It only provides the transmission management of the DLS_User data.

7.2.4 DLL services

The services which DLL provides to DLS_User defined in ISO/IEC 8802-3 are applied without any changing.

7.2.5 Transaction between DLL and PhL

The transactions between DLL and PhL defined in ISO/IEC 8802-3 are applied without any changing.

7.3 EPA communication scheduling procedure

7.3.1 General

Within an EPA micro-segment, the communication procedure is repeated. The time to complete a communication procedure is called communication macrocycle and marked as T. Each communication macrocycle (T) is divided into two phases, periodic packet transferring phase (Tp) and non-periodic packet transferring phase (Tn) (see Figure 8).

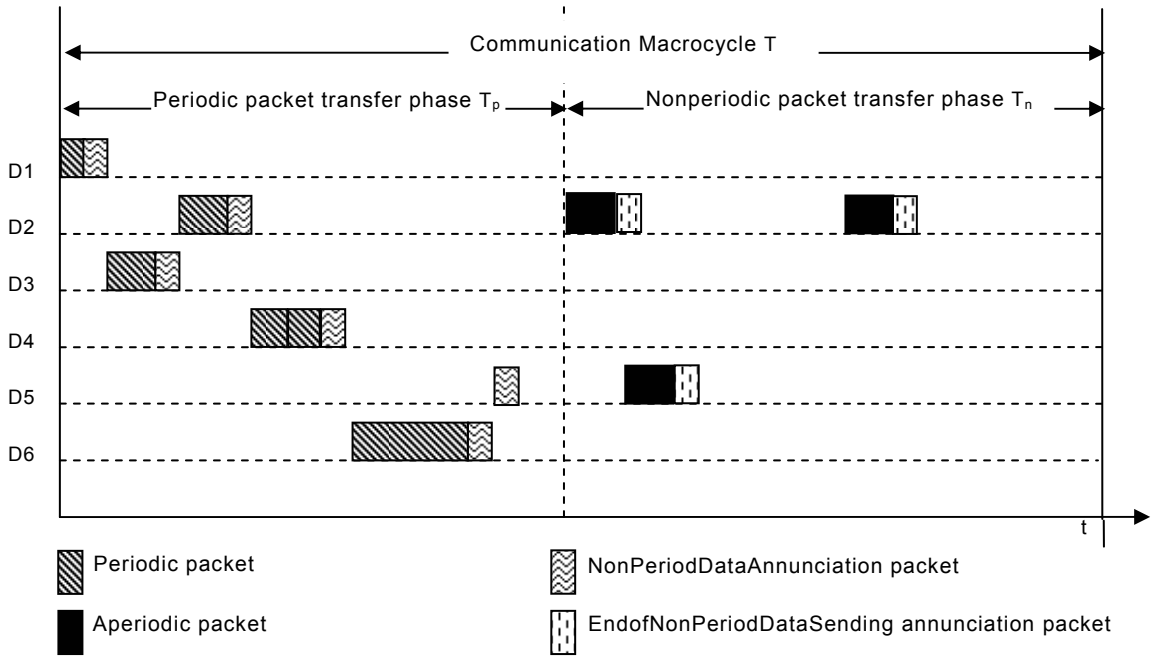


Figure 8 – time-sharing communication scheduling

In periodic packet transferring phase (Tp), the ECSME of each EPA device delivers periodic DLS_User data to local DLE in a configured order. Local DLE will packet and send the data on the network after it receives them.

Periodic DLS_User data contain the data relevant to process parameters, such as measurement and control data which need to be transmitted periodically in the control loop, or the input/output data which need to be updated cyclically between function blocks. Periodic DLS_User data has the highest priority to be sent.

In non-periodic packet transferring phase (Tn), the ECSME of each EPA device delivers non-periodic DLS_User data to local DLE according to their priority and even local IP address. Local DLE will packet and send the data on the network after it receives them.

Non-periodic DLS_User data contain the data which don't need to send out in every macrocycle. That is, non-periodic DLS_User data are not produced cyclically. In EPA systems, non-periodic DLS_User data are the primitives of program upload/download, variable read/write, event notification, trend report and RARP, HTTP, FTP, TFTP, ICMP, IGMP application data etc. In other cases, the primitives of SNTP and DHCP services are regarded as non-periodic DLS_User data.

Non-periodic data are sent out according to their priority, local IP address in time-available way. That is,

a) if time is available, the non-periodic packet with high priority is sent first;

b) if two or more non-periodic packets in local device have identical priority, the first-produced one is sent first;

c) if two or more non-periodic packets distributed in different devices, those located in the devices with smaller IP address is sent out first.

7.3.2 Scheduling procedure

The mechanism of ECSME can be described in four states: Standby, Ready, PeriodicDataSending and NonPeriodicDataSending. The transitions between these four states are shown in Figure 9 and Table 4.

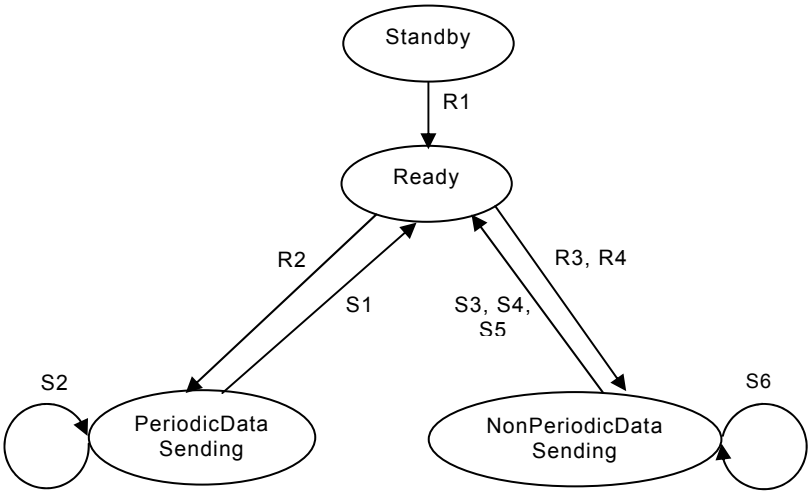


Figure 9 – States transitions of ECSME

7.3.2.1 State description

7.3.2.1.1 Standby

After powered up, EPA device shall verify all necessary operation parameters (see clause 0). If these parameters have not been configured, ECSME enters the *Standby* state until the device is configured.

In *Standby* state, ECSME delivers data between DLS and DLS_User directly without any processing and buffering.

After the configuration of EPA Communication Scheduling Management Class of local device is finished, ECSME will enter the *Ready* state automatically, and calculate the time offset from local current time to the starting time of a communication macrocycle (T) according to the following formula:

Time offset = MOD (Local current time, T)

Then, the starting time of its first communication macrocycle (T) is:

Starting time = Current Time + Time offset

7.3.2.1.2 Ready

In *Ready* state,

a) When receiving DLS_User data from DLS_User, ECSME shall put it into responding unsent message queue according to its transmission priority;

NOTE 1 ECSME shall establish different queues for DLS_User data to be sent with different priorities.

NOTE 2 The queue with priority 0 shall be managed according to the value of SendingTimeOffset corresponding to each DLS_User data.

b) When receiving DLPDU from DLE,

1) if the DLPDU is a NonPeriodicDataAnnunciation message, ECSME shall maintain local non-periodic packet transferring management list using the remote IP addresses and transmission priorities from the DLPDUs;

2) otherwise, ECSME shall deliver the DLPDU to DLS_User directly.

7.3.2.1.3 PeriodicDataSending

When the following event of

MOD (Current time, T) = SendingTimeOffset of periodic packets

is detected (see clause 8.3.2.3.11), the state of ECSME shall be changed into *PeriodicDataSending*.

In *PeriodicDataSending* state, ECSME deliver the DLS_User data to DLE in order, and DLE transfer them on the network through PhLE,then it shall transmit NonPeriodicDataAnnunciation message to tell remote the priority of non-periodic packet to be sent in local device. After doing above, the state of ECSME is changed into *Ready*.

7.3.2.1.4 NonPeriodicDataSending

When the following event of

MOD (Current time, T) = NonPeriodicDataTransferOffset

is detected (see clause 0), or local ECSME receive the EndofNonPeriodDataSending message from remote device, ECSME enters *NonPeriodicDataSending* state.

In this state,

a) if there is no non-periodic DLS_User data – whose priority is not equal to 0 – to be sent in local device, ECSME takes step g), otherwise takes step b);

b) If the priority of the first non-periodic DLS_User data in local device is lower than those in local non-periodic packet transferring management list for remote devices, ECSME takes step g), otherwise takes step c);

c) If the priority of the first non-periodic DLS_User data in local device is higher than those in remote devices, ECSME takes step e), otherwise takes next step;

d) When the priority of the first non-periodic DLS_User data in local device is equal to at least one of those in remote devices, and this priority is the highest, and if the IP address of local device is larger than that of all other remote devices with the same priority, ECSME takes step g), otherwise takes next step;

e) If the remaining time of current communication macrocycle is enough to transmit it, ECSME delivers the non-periodic DLS_User data to DLE for sending to the network, then ECSME goes to next setp, otherwise ECSME reserves the data for next transmission, goes to step g);

f) If there is more non-periodic packet to be transmitted, ECSME goes to step b);

g) If local device has transmitted at least one non-periodic DLS_User data, ECSME transmits EndofNonPeriodicDataSending annunciation message on the network to notify remote devices;

h) The state of ECSMA is changed into *Ready* state.

7.3.2.2 State table

The state transitions of ECSME are illustrated in Table 4.

Table 4 – ECSME state transitions

#	Current state	Event or condition => actions	Next state
S1	PeriodicDataSending	Epnon-periodicDataSendingSuc()=1 => EpaNonPeriodicDataAnnunciation()	Ready
S2	PeriodicDataSending	Epnon-periodicDataSendingSuc()=2 => Epnon-periodicDataSending()	PeriodicDataSending
S3	NonPeriodicDataSending	EpaNonPeriodicDataSendingSuc()=3 || (EpaFirstNonPeriodicDataSending()=TRUE && (EpaNonPeriodicDataPriority()=FALSE || EpaNonPeriodicDataTimeEnough()=FALSE)) => (no actions taken)	Ready
S4	NonPeriodicDataSending	EpaNonPeriodicDataSendingSuc()=1 => EpaEndofNonPeriodicDataSending()	Ready
S5	NonPeriodicDataSending	EpaNonPeriodicDataSendingSuc()=2 && (EpaNonPeriodicDataPriority()=FALSE || (EpaNonPeriodicDataPriority()=TRUE && EpaNonPeriodicDataTimeEnough()=FALSE)) => EpaNonPeriodicDataAnnunciation()	Ready
S6	NonPeriodicDataSending	EpaNonPeriodicDataPriority()=TRUE && EpaNonPeriodicDataTimeEnough()=TRUE => EpaNonPeriodicDataSending()	NonPeriodicDataSending
R1	Standby	EpaIsDeviceConfigured()=TRUE => EpaCountOffsetTime()	Ready
R2	Ready	EpaDataSendingTiming()=1 => Epnon-periodicDataSending()	PeriodicDataSending
R3	Ready	EpaDataSendingTiming()=2 => (no actions taken)	NonPeriodicDataSending
R4	Ready	EpaRecEndofNonPeriodicDataSending() =TRUE => (no actions taken)	NonPeriodicDataSending

7.3.2.3 Function descriptions

Table 5 through Table 17 describe the functions referenced by the preceding state transitions table.

7.3.2.3.1 Epnon-periodicDataSendingSuc( )

The description of function Epnon-periodicDataSendingSuc() is illustrated in Table 5.

Table 5 – Epnon-periodicDataSendingSuc() description

Name	**Epnon-periodicDataSendingSuc( )**	**Used in**	**ECSME**
Input		Output	
(none)		data of Unsigned8 data type	
Function			
Checks whether all periodic packets have been transmitted in current Communication Macrocycle: if yes, returns the value of 1; if no, returns the value of 2; otherwise, returns the value of zero.			

7.3.2.3.2 EpaNonPeriodicDataAnnunciation( )

The description of function EpaNonPeriodicDataAnnunciation() is illustrated in Table 6.

Table 6 – EpaNonPeriodicDataAnnunciation() description

Name	**EpaNonPeriodicDataAnnunciation( )**	**Used in**	**ECSME**
Input		Output	
(none)		(none)	
Function			
Transmits NonPeriodicDataAnnunciation packet.			

7.3.2.3.3 Epnon-periodicDataSending( )

The description of function Epnon-periodicDataSending() is illustrated in Table 7.

Table 7 – Epnon-periodicDataSending() description

Name	**Epnon-periodicDataSending( )**	**Used in**	**ECSME**
Input		Output	
(none)		(none)	
Function			
Transmits periodic packet.			

7.3.2.3.4 EpaNonPeriodicDataSendingSuc( )

The description of function EpaNonPeriodicDataSendingSuc() is illustrated in Table 8.

Table 8 – EpaNonPeriodicDataSendingSuc() description

Name	**EpaNonPeriodicDataSendingSuc( )**	**Used in**	**ECSME**
Input		Output	
(none)		data of Unsigned8 data type	
Function			
Checks whether all non-periodic packets have been transmitted in current Communication Macrocycle, if yes, returns the value of 1; if no, renturns the value of 2; if none of non-periodic packet needs to be transmitted, returns the value of 3; otherwise, returns the value of zero.			

7.3.2.3.5 EpaFirstNonPeriodicDataSending( )

The description of function EpaFirstNonPeriodicDataSending() is illustrated in Table 9.

Table 9 – EpaFirstNonPeriodicDataSending() description

Name	EpaFirstNonPeriodicDataSending( )	Used in	ECSME
Input		Output	
(none)		TRUE or FALSE	
Function			
Checks whether it's the first time to transmit non-periodic packet in current Communication Macrocycle. if yes, returns TRUE, otherwise returns FALSE.			

7.3.2.3.6 EpaNonPeriodicDataPriority( )

The description of function EpaNonPeriodicDataPriority() is illustrated in Table 10.

Table 10 – EpaNonPeriodicDataPriority() description

Name	EpaNonPeriodicDataPriority( )	Used in	ECSME
Input		Output	
(none)		TRUE or FALSE	
Function			
Checks whether the priority of non-periodic packet that local device is ready to transmit is the highest in local non-periodic packet transfer management list. If yes, returns TRUE, otherwise returns FALSE.			
NOTE If the priority of non-periodic packet that local device is ready to transmit is equal to that of one remote device or several remote devices, moreover the priority is the highest in the non-periodic packet transfer management list, however the IP address of local device is larger than that of all other remote devices with the same priority, then still returns FALSE.			

7.3.2.3.7 EpaNonPeriodicDataTimeEnough( )

The description of function EpaNonPeriodicDataTimeEnough() is illustrated in Table 11.

Table 11 – EpaNonPeriodicDataTimeEnough() description

Name	EpaNonPeriodicDataTimeEnough( )	Used in	ECSME
Input		Output	
(none)		TRUE or FALSE	
Function			
Checks whether the remaining time in current Communication Macrocycle is enough to transmit non-periodic packet as well as its annunciation message or not. if yes, returns TRUE, otherwise returns FALSE.			

7.3.2.3.8 EpaNonPeriodicDataSending( )

The description of function EpaNonPeriodicDataSending() is illustrated in Table 12.

Table 12 – EpaNonPeriodicDataSending() description

Name	EpaNonPeriodicDataSending( )	Used in	ECSME
Input		Output	
(none)		(none)	
Function			
Transmits non-periodic packet.			

7.3.2.3.9 EpaEndofNonPeriodicDataSending( )

The description of function EpaEndofNonPeriodicDataSending() is illustrated in Table 13.

Table 13 – EpaEndofNonPeriodicDataSending() description

Name	EpaEndofNonPeriodicDataSending( )	Used in	ECSME
Input		Output	
(none)		(none)	
Function			
Transmits EndofNonPeriodicDataSending annunciation packet.			

7.3.2.3.10 EpaIsDeviceConfigured( )

The description of function EpaIsDeviceConfigured() is illustrated in Table 14.

Table 14 – EpaIsDeviceConfigured() description

Name	EpaIsDeviceConfigured( )	Used in	ECSME
Input		Output	
(none)		TRUE or FALSE	
Function			
After local device is powered-up, this function is called to check whether the device has been configurated or not. If yes, returns TRUE, otherwise returns FALSE.			

7.3.2.3.11 EpaCountOffsetTime( )

The description of function EpaCountOffsetTime() is illustrated in Table 15.

Table 15 – EpaCountOffsetTime() description

Name	EpaCountOffsetTime( )	Used in	ECSME
Input		Output	
(none)		Data of Unsigned32 type	
Function			
Calculates MOD(local current time ,T), returns the remainder of Unsigned32 data type, its unit is millisecond.			

7.3.2.3.12 EpaDataSendingTiming( )

The description of function EpaDataSendingTiming() is illustrated in Table 16.

Table 16 – EpaDataSendingTiming() description

Name	EpaDataSendingTiming( )	Used in	ECSME
Input		Output	
(none)		Data of Unsigned8 type	
Function			
This function calls EpaCountOffsetTime() to get the remainder. If the remainder is equal to the value of SendingTimeOffset, returns the value of 1; if the remainder is equal to the value of NonPeriodicDataTransferOffset, returns the value of 2; otherwise, returns the value of zero.			

7.3.2.3.13 EpaRecEndofNonPeriodicDataSending( )

The description of function EpaRecEndofNonPeriodicDataSending() is illustrated in Table 17.

Table 17 – EpaRecEndofNonPeriodicDataSending() description

Name	EpaRecEndofNonPeriodicDataSending( )	Used in	ECSME
Input		Output	
(none)		TRUE or FALSE	
Function			
Check whether an EndofNonPeriodicDataSending annunciation packet from remote device is reveived or not. If yes, returns TRUE, otherwise returns FALSE.			

7.3.3 Priority

Six-level priorities for packet transmission are defined in this specification, namely 0, 1, 2, 3, 4, 5, and the value of 0 represents the highest priority (see 8.4).

This specification adopts a registered value of 0x88BC assigned by IEEE for LENGTH/TYPE field in Ethernet frame to identify EPA packet (see Figure 10).

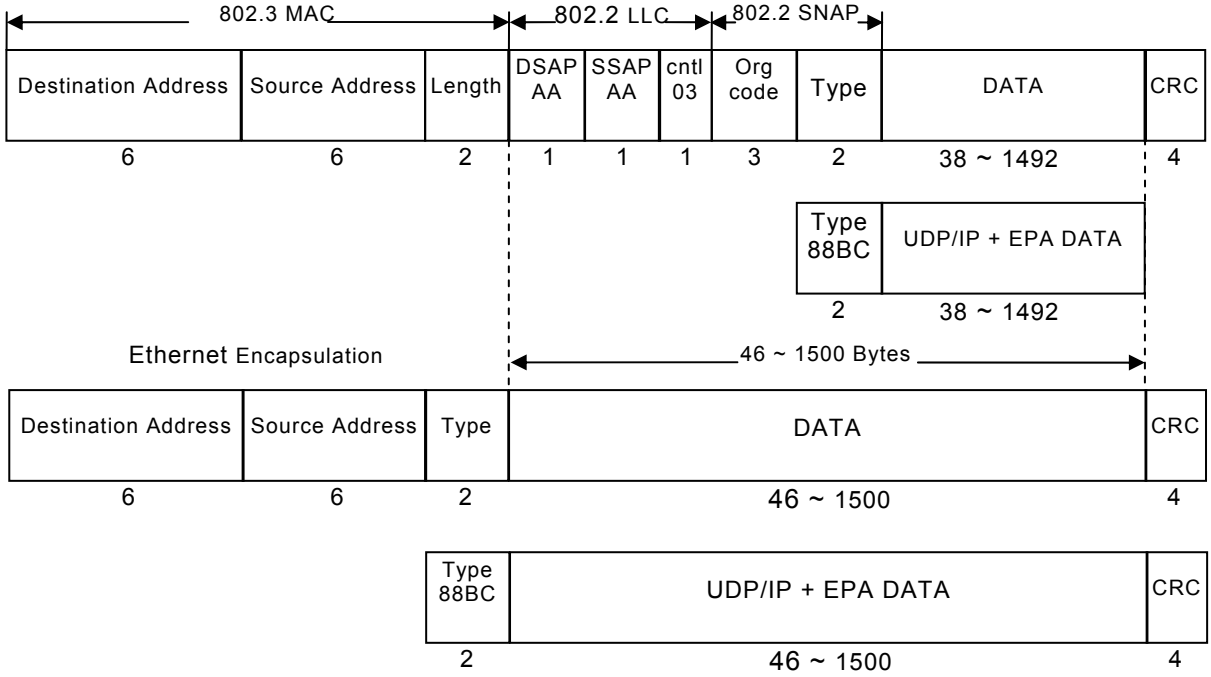


Figure 10 – EPA packet identifier

7.3.4 Time synchronization

The time in all EPA devices shall be synchronized for the purpose of communication scheduling (see clause 8).

7.4 ECSME PDU structure

7.4.1 NonPeriodicDataAnnunciation PDU

The format of NonPeriodicDataAnnunciation PDU is described in Figure 11.

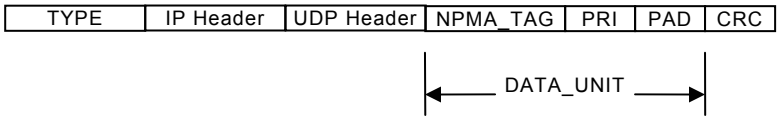


Figure 11 – Format of NonPeriodicDataAnnunciation PDU

TYPE

Protocol type, the length of this field is two bytes. Its value is 0x88BC.

IP Header

IP header, the length of this field is 20 bytes referring to RFC 791, Internet Protocol (IP).

UDP Header

UDP header, the length of this field is 8 bytes referring to RFC 768, User Datagram Protocol (UDP).

NPMA_TAG

NonPeriodicDataAnnunciation PDU identifier, the length of this field is one byte. Its value is 0x20.

PRI

Priority, the length of this field is one byte, the value of this field indicates the priority of next non-periodic packet, if the value of this field is equal to 0xFF, it's illustrated that no non-periodic packet needs to be transmitted.

PAD

Pad octets, the length of this field is 44 bytes, all values of this field are set 0x20.

CRC

Cyclic Redundancy Check value, the length is 4 bytes.

DATA_UNIT

Data unit, the length of this field is 46 bytes.

7.4.2 EndofNonPeriodicDataSending PDU

The format of EndofNonPeriodicDataSending PDU is described in Figure 12.

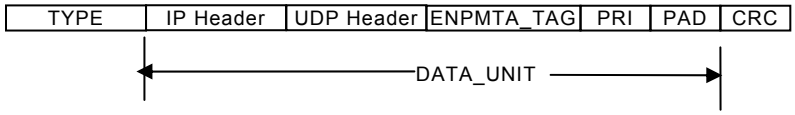


Figure 12– Format of EndofNonPeriodicDataSending PDU

TYPE

Protocol type, the length of this field is two bytes. Its value is 0x88BC.

IP Header

IP header, the length of this field is 20 bytes referring to RFC 791, Internet Protocol (IP).

UDP Header

UDP header, the length of this field is 8 bytes referring to RFC 768, User Datagram Protocol (UDP).

ENPMTA_TAG

EndofNonPeriodicDataSending PDU identifier, the length of this field is one byte. Its value is 0x21.

PRI

Priority, the length of this field is one byte, indicating the priority of the unsent non-periodic packet. The value of 0xFF indicates that no non-periodic packet needs to be transmitted.

PAD

Pad octets, the length of this field is 44 bytes, all values of this field are set 0x20.

CRC

Cyclic Redundancy Check value, the length is 4 bytes.

DATA_UNIT

Data unit, the length of this field is 46 bytes.

7.4.3 EPA PDU

The format of EPA PDU is described in Figure 13.

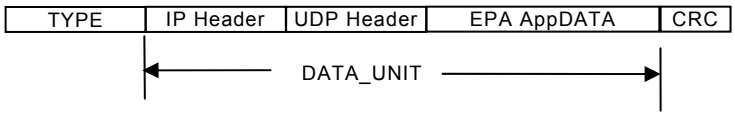


Figure 13– Format of EPA PDU

TYPE

Protocol type, the length of this field is two bytes. Its value is 0x88BC.

IP Header

IP header, the length of this field is 20 bytes referring to RFC 791, Internet Protocol (IP).

UDP Header

UDP header, the length of this field is 8 bytes referring to RFC 768, User Datagram Protocol (UDP).

EPA AppDATA

EPA application layer primitives.

CRC

Cyclic Redundancy Check, the length of this field is 4 bytes.

DATA_UNIT

Data unit, the minimum length of this field is 46 bytes.

7.5 Encoding of ECSME Packet

7.5.1 Encoding of NonPeriodicDataAnnunciation message

The encoding of NonPeriodicDataAnnunciation message is described in Table 18.

Table 18 – NonPeriodicDataAnnunciation message Encoding

No.	Parameter name	Data type	Octet offset	Byte length	Description
1	Type	Unsigned16	0	2	Protocol type, the value is 0x88BC
2	IP Header	OctetString	2	20	IP header, referring to RFC 791, Internet Protocol (IP)
3	UDP Header	OctetString	22	8	UDP header, referring to RFC 768, User Datagram Protocol(UDP)
2	NPMA_TAG	Unsigned8	30	1	NonPeriodicDataAnnunciation PDU identifier, the value is 0x20
3	PRI	Unsigned8	31	1	Priority, indicates the priority of next non-periodic data to be transmitted, the value of 0xff indicates no non-periodic data needs to be transmitted.
4	PAD	OctetString	32	44	Pad octets, all values are 0x20
5	CRC	Unsigned32	76	4	Cyclic Redundancy Check value

7.5.2 Encoding of EndofNonPeriodicDataSending message

The encoding of EndofNonPeriodicDataSending Message is described in Table 19.

Table 19 – EndofNonPeriodicDataSending message Encoding

No.	Parameter name	Data type	Octet offset	Byte length	Description
1	Type	Unsigned16	0	2	Protocol type,the value is 0x88BC
2	IP Header	OctetString	2	20	IP header, referring to RFC 791, Internet Protocol (IP)
3	UDP Header	OctetString	22	8	UDP header, referring to RFC 768, User Datagram Protocol(UDP)
2	ENPMTA_TAG	Unsigned8	30	1	EndofNonPeriodicDataSending PDU identifier, the value is 0x21
3	PRI	Unsigned8	31	1	Priority, indicates the priority of next non-periodic data to be transmitted, the value of 0xff indicates no non-periodic data needs to be transmitted.
4	PAD	OctetString	32	44	Pad octets, all values are 0x20
5	CRC	Unsigned32	76	4	Cyclic Redundancy Check value

7.5.3 Encoding of EPA message

The encoding of EPA message is described in Table 20.

Table 20 – EPA message Encoding

No.	Parameter name	Data type	Octet offset	Byte length	Description
1	TYPE	Unsigned16	0	2	Protocol type, the value is 0x88BC
2	IP Header	OctetString	2	20	IP header, referring to RFC 791, Internet Protocol (IP)
3	UDP Header	OctetString	22	8	UDP header, referring to RFC 768, User Datagram Protocol (UDP)
4	EPA AppDATA	OctetString	30	N	The data of EPA application layer, the minimum length is 46 bytes
5	CRC	Unsigned32	30+N	4	Cyclic Redundancy Check value

8 EPA application layer service definition

8.1 Concepts

This subclause describes fundaments of the EPA Application Layer.

8.1.1 Overview

This subclause introduces EPA Application Layer architecture and services. Application Layer services provide the support to EPA management system and User Layer AP.

8.1.2 Architectural relationships

8.1.2.1 Relationship to the application layer of the OSI basic reference model

The functions of the EPA Application Layer have been described according to OSI reference model principles. However, its architectural relationship to the lower layers is different, as shown in Figure 14.

a) The EPA Application Layer includes OSI functions together with extensions to cover time-critical requirements. The OSI Application Layer Structure standard (ISO/IEC 9545) was used as a basis for specifying the EPA Application Layer.

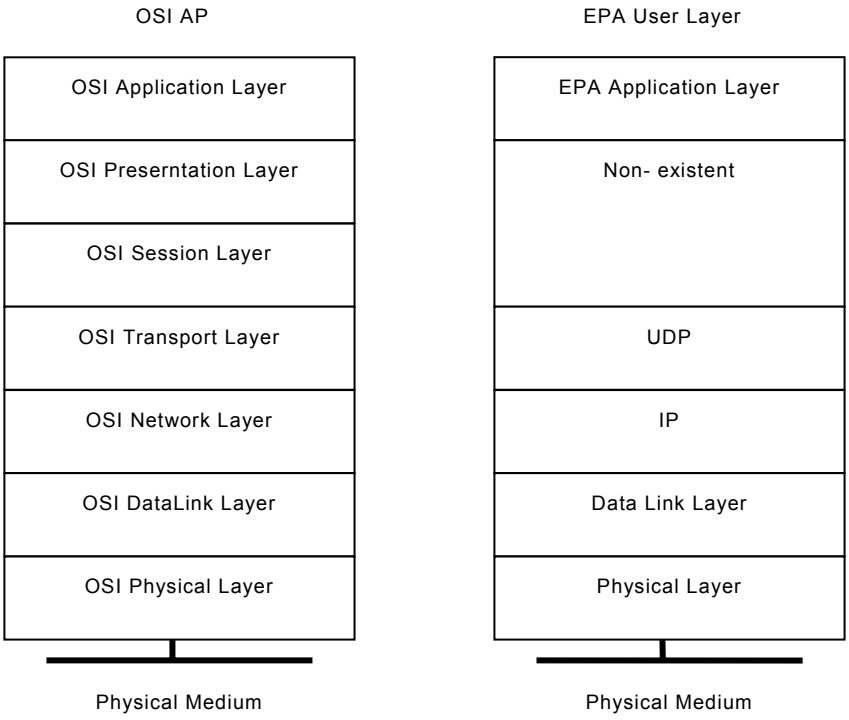


Figure 14 – Relationship to the OSI basic reference model

b) The EPA Application Layer directly uses the services of the underlying layer. The underlying layer may be the transport layer defined in RFC 768, User Datagram Protocol (UDP). When using the underlying layer, the EPA Application Layer may provide functions normally associated with the OSI Middle Layers for proper mapping onto the underlying layer.

8.1.2.2 Relationship to other EPA entities

8.1.2.2.1 General

The EPA Application Layer architectural relationships, as illustrated in Figure 15, have been designed to support the interoperability needs of time-critical systems distributed within the EPA environment.

Within this environment, the EPA Application Layer provides communication services to time-critical and non-time-critical applications located in EPA devices.

In addition, the EPA Application Layer uses Ethernet/UDP/IP protocols to transfer its application layer protocol data units.

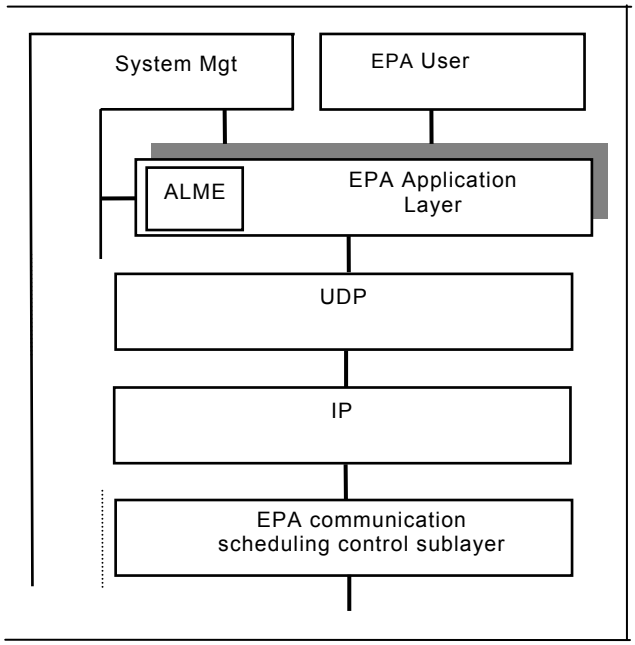


Figure 15 – Architectural positioning of the EPA Application Layer

8.1.2.2.2 Use of UDP/IP

The EPA Application Layer provides network access to APs. It interfaces directly to the UDP/IP for transfer of its APDUs.

8.1.2.2.3 Support to EPA applications

IEC 61158-5, 4.2.2.3 applies.

8.1.2.2.4 Support to system management

IEC 61158-5, 4.2.2.4 are applies.

8.1.2.2.5 Access to EPA application layer management entity

IEC 61158-5, 4.2.2.5 apply.

8.1.3 EPA application layer structure

8.1.3.1 Overview

The structure of the EPA application layer is a refinement of the OSI Application Layer Structure (ISO/IEC9545). As a result, the organization of this subclause is similar to that of ISO/IEC 9545.Certain concepts presented here have been refined from ISO/IEC 9545 for the EPA environment.

The EPA application layer differs from the other layers of OSI in two principal aspects:

a) OSI defines a single type of application layer communications channel, the association, to connect APs to each other. The EPA application layer defines the Application Relationship (AR), of which there are several types, to permit application processes (APs) to communicate with each other.

b) The EPA application layer uses Ethernet/UDP/IP to transfer its PDUs and not the presentation layer. Therefore, there is no explicit presentation context available to the EPA application layer.

8.1.3.2 Fundamental concepts

IEC 61158-5, 4.3.2 are applies.

8.1.3.3 EPA application processes

IEC 61158-5, 4.3.3 are applies.

8.1.3.4 Application process objects

IEC 61158-5, 4.3.4 are applies.

8.1.3.5 Application entities

See IEC 61158-5, 4.3.5 applies.

8.1.3.6 EPA application service elements

IEC 61158-5, 4.3.6 are applies.

8.1.3.7 Application relationships

IEC 61158-5, 4.3.7 are applies.

8.1.4 EPA application layer naming and addressing

IEC 61158-5, 4.4 are applies.

8.1.5 Architecture summary

IEC 61158-5, 4.5 are applies.

8.1.6 EPA application layer services procedures

IEC 61158-5, 4.6 are applies.

8.2 Data Type ASE

IEC 61158-5, clause 5 applies.

8.3 Communication model specification

8.3.1 EPA AE

EPA AE is composed of EPA System Management Entity, EPA Application Access Entity and EPA Socket Mapping Entity, as shown in Figure 16.

EPA System Management Entity and Services can support various management operations, including system management ASE, AR ASE, etc.

EPA Application Access Entity provides an interface for data communication between user application process, which is composed of domain ASE, variable ASE and event ASE.

EPA Socket Mapping Entity provides an interface for EPA Application Access Entity, System Management Entity and UDP/IP, which is composed of EPA Socket Mapping ASE.

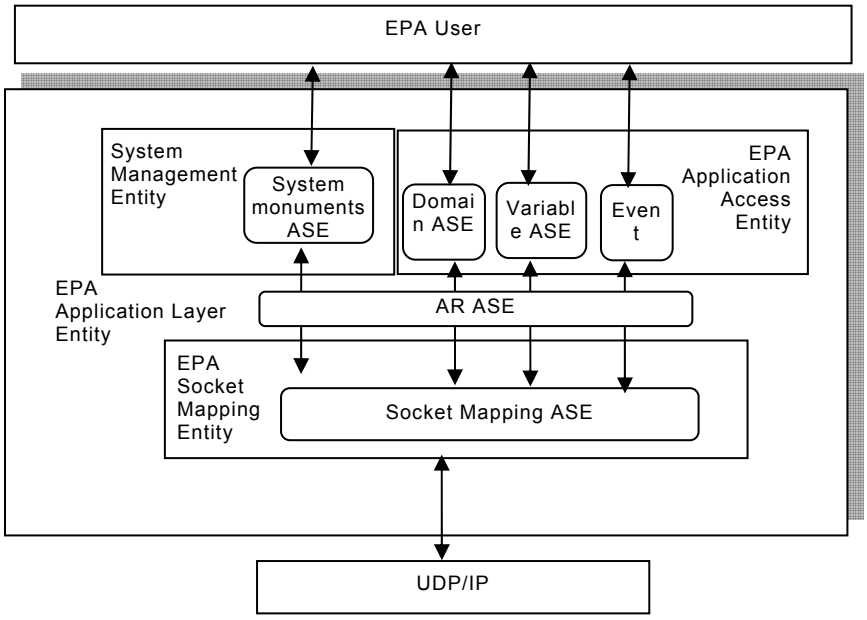


Figure 16 – EPA Application Layer Entity

8.3.2 EPA System Management ASE

8.3.2.1 Overview

EPA System Management is used to integrate several devices in EPA network into a harmonized communication system. EPA System Management supports the function of device identification, address assignment, object location, clock synchronization and EPA link management.

8.3.2.1.1 Device identification

An EPA device can be identified by its physical device tag, device ID, device redundancy number (mutually redundancy devices).

8.3.2.1.1.1 Device ID

Device ID identifies a unique device. A unique Device ID is assigned to each individual device by the manufacturer. It is visible in EPA system management entity, but cannot be modified.

8.3.2.1.1.2 PD_Tag

For the purposes of device configuration, each field device has a physical device tag (PD_Tag). In an EPA system, PD_Tag is unique to identifier a physical device. Mutually redundancy device has the same PD_Tag.

8.3.2.1.1.3 Redundancy Number

Mutually redundancy devices distinguish each through the different redundancy number.

8.3.2.1.2 Address assignment

The IP address of each EPA device can be statically appointed or dynamically assigned through Dynamic Host Configuration Protocol (DHCP). Once the IP address is assigned, it shall be written into non-volatile memory as permanent information until device is reconfigured.

When the IP address confliction with other device is detected, the state of local device shall be changed into *No Address*, waiting to reset its IP address.

8.3.2.1.3 Object Location

An EPA physical device can be located by IP address or PD_Tag. An FB instance in an EPA device can be located by FB_Tag or FB identifier AppID, which are unique in each EPA device, but not unique in the whole EPA network. Each parameter object of an FB instance can be located by parameter index, ObjectID, which is unique in each FB instance, but not unique in the whole device.

EPA System Management entity provides EM_FindTagQuery service to query the IP address of an EPA device using PD_Tag. It also provides EM_GetDeivceAttibute service to get the PD_Tag or other relevant information of an EPA physical device.

EM_FindTagQuery service request primitive can be distributed in EPA network with unicast or multicast mode. The physical device receiving this request primitive will check its local PD_Tag. If local PD_Tag matches that in EM_FindTagQuery service request primitive, it will report its IP address, Device ID using EM_FindTagReply service.

EM_GetDeivceAttibute service request primitive can be delivered to EPA network using unicast mode.

8.3.2.1.4 Adding or deleting a device

After the power-on and initialization, if an EPA device has no IP address, it enters *No Address* state to wait for IP address assignment. After IP address assignment, the device changes its state into *Unconfigured* and broadcasts a device annunciation message using EM_DeviceAnnunciation service.

Once the user application receives the device annunciation message, it can configure the device attributes using EM_SetDeviceAttribute service. After configuration, the device can operate in *Configured* state. The configuration information is written in non-volatile memory as permanent information until device is reconfigured.

When powered-on again, the device can recover all configuration information and operate in *Configured* state automatically.

8.3.2.1.5 Clock synchronization

The time of each EPA device shall be synchronized. Either SNTP protocol defined in RFC2030 or IEC 61588 can be used according to practical application.

At each time of synchronization, local device sends a clock synchronization request primitive to time server at a configured time interval and keep synchronization with system time according to the response from time server.

The details of time synchronization is not specified in this specification (see RFC 2030 and IEC 61588).

8.3.2.1.6 EPA Link Management

The access path between FBs is defined as EPA link object. The EPA link object specifies the communication relationship between the input/output of FBs. EPA link objects shall be configured using configuration program.

Through a configured EPA link object, an FB in an EPA device can determine where to release its output data or where to receive updating input data.

8.3.2.2 EPA Management Information Base

All the management objects using in EPA system management entity are organized in EPA Management Information Base (EPA MIB). EPA MIB is a two-dimension table as shown in Figure 21. Each object has a unique identifier ObjectID.

As a special function block, EPA management information base has its default value of 0 for application identifier AppID for the purpose of reading/writing the objects. That is, other function block instance shall have the value of more than 0 for application identifier AppID.

Table 21 – EPA MIB

EPA Object	Object ID	Illustration
EPA MIB Header	1	EPA device Management Information Base header object
EPA Device Descriptor	2	EPA device descriptor object
Time Synchronization	3	Time Synchronization Object
Max Response Time	4	Confirmed Service Max Response Time
Communication Schedule Management	5	Communication Schedule Management Object
Device Application information	6	Device Application information Object
FB Application information Header	7	Function Block Application information Header
Link Object Header	8	Link Object Header
Domain Application Object Header	9	Domain Application Object Header
……		
FB Application information 1	2000	Function Block Application information 1
FB Application information 2	2001	Function Block Application information 2
……	Increased number in turn	
Domain Application Object 1	4000	Domain Application Object 1
Domain Application Object 2	4001	Domain Application Object 2
……	Increased number in turn	
Link Object 1	5000	Link Object 1
Link Object 2	5001	Link Object 2
……	Increased number in turn	

8.3.2.3 System management model specification

8.3.2.3.1 EPA MIB Header Class

EPA MIB Header Class represents the version number of EPA device management information base.

8.3.2.3.1.1 Formal model

EPA MIB header class is presented as follows:

EPA ASE: **SYSTEM MANAGEMENT ASE**
CLASS: **EPA MIB HEADER**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:
1. (m) Key Attribute: Object ID
2. (m) Attribute: SMIB Revision Number

SERVICES:
1. (o) OpsService: Read

8.3.2.3.1.2 Attributes

Object ID

This attribute identifies EPA MIB Header object in EPA MIB. The value for EPA device MIB Header object is 1.

SMIB Revision Number

This attribute indicates the revision number of EPA MIB, which is assigned by user.

8.3.2.3.1.3 Services

Read

The service allows user to read EPA device MIB Header Class attributes.

Read service is defined in 8.3.5.3.1.

8.3.2.3.2 Device Descriptor Class

This class specifies the basic attributes of an EPA device, i.e. Device ID, Device Type, Device Tag, IP address etc. Through EPA device management services, user application program can get or set the basic attributes of this device.

8.3.2.3.2.1 Formal model

EPA ASE:		**SYSTEM MANAGEMENT ASE**	
CLASS:		**EPA DEVICE DESCRIPTOR**	
CLASS ID:		**Not Used**	
PARENT CLASS:		**TOP**	
ATTRIBUTES:			
1.	(m)	Key Attribute:	Object ID
2	(m)	Attribute:	Reserved
3.	(m)	Key Attribute:	Device ID
4.	(m)	Attribute:	PD_Tag
5.	(m)	Attribute:	Active IP Address
6.	(m)	Attribute:	Device Type
7.	(m)	Attribute:	Status
8.	(m)	Attribute:	Device Version
9.	(m)	Attribute:	Annunciation Interval
10.	(m)	Attribute:	Annunciation Version Number
11.	(m)	Attribute:	Device Redundancy State
12.	(m)	Attribute:	Device Redundancy Number
13	(m)	Attribute:	LAN Redundancy Port
14.	(m)	Attribute:	Max Redundancy Number
15.	(m)	Attribute:	Duplicate Tag Detected
SERVICES:			
1.	(o)	OpsService:	EM_FindTagQuery
2.	(o)	OpsService:	EM_FindTagReply
3.	(o)	OpsService:	EM_SetDeviceAttribute
4.	(o)	OpsService:	EM_GetDeviceAttribute
5.	(o)	OpsService:	EM_ClearDeviceAttribute
6.	(o)	OpsService:	EM_DeviceAnnunciation

8.3.2.3.2.2 Attributes

Object ID

This attribute identifies Device Descriptor object in EPA MIB. Its value is 2 for Device Descriptor object.

Reserved

This field is reserved.

Device ID

This attribute specifies the vendor specific EPA physical Device. It is set by manufacturer.

PD_Tag

This attribute specifies the site administered name (tag) assigned to the device.

Active IP Address

This attribute indicates current operable IP address.

Device Type

This attribute specifies the type of device and its capability. It is set by manufacturer.

Status

This attribute indicates the status of local device. There are three optional status:

0——No address;
1——Unconfigured;
2——Configured or operable.

Device Version

This attribute is the version number of the device. It is assigned by the vendor when it is produced.

Annunciation Interval

This attribute specifies the milliseconds between annunciation messages. Its default value is 15000 (15 seconds).

Annunciation Version Number

This attribute is the version number that represents the composite state of the static information in the applications in the device. Each time there is change to the static data of any application in the device, this version number is incremented by 1. Its value initializes to zero if the device is initialized.

Device Redundancy State

This attribute specifies the actual device redundancy state of the device. 0 indicates the device is active while 1 represents that the device is backup. If the device is not participating in redundancy, its value is 0xFF.

Device Redundancy Number

This attribute specifies the device redundancy number. Once the active device fails, the minimum number of backup device firstly become active device.

Max Redundancy Number

This attribute specifies the max redundancy number.

LAN Redundancy Port

This attribute specifies the port which is used to exchange data between redundancy devices.

Duplicate Tag Detected

The attribute specifies whether the device PD_Tag conflict with other device PD_Tag in network or not. That value is TRUE means occurring conflict.

8.3.2.3.2.3 Services

EM_FindTagQuery

This optional service allow user to find device information according to device PD_Tag.

EM_FindTagReply

This optional service is used to reply the device which sends EM_FindTagQuery request.

EM_SetDeviceAttribute

This optional service allows user to set the EPA device attributes.

EM_GetDeviceAttribute

This optional service allows user to get the EPA device attributes.

EM_ClearDeviceAttribute

This optional service allows user to clear the EPA device attributes..

EM_DeviceAnnunciation

This service is used for an EPA device to announce its presence on the network.

8.3.2.3.3 Time Synchronization Class

In order implemented the clock synchronization of device in EPA network, each EPA device maintains a local Current Time. Through time synchronization, the local current can be synchronized with time server in requested synchronization precision.

8.3.2.3.3.1 Formal model

This class is presented as follows:

EPA ASE: **SYSTEM MANAGEMENT ASE**
CLASS: **TIME SYNCHRONIZATION**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1.	(m)	Key Attribute:	Object ID
2.	(m)	Attribute:	Reserved
3.	(m)	Attribute:	Primary Time Server
4.	(m)	Attribute:	Secondary Time Server
5.	(m)	Attribute:	Time Request Timeout
6.	(m)	Attribute:	Time Request Interval
7.	(m)	Attribute:	Capable Time Sync Class
8.	(m)	Attribute:	Target Time Sync Class
9.	(m)	Attribute:	Current Time
10.	(m)	Attribute:	Standard Time Difference

SERVICES:

1.	(o)	OpsService:	Read
2.	(o)	OpsService:	Write
3.	(o)	OpsService:	Clock Synchronization service defined in RFC 2030 or IEC 61588:2004.

8.3.2.3.3.2 Attributes

Object ID

This attribute identifies Time Synchronization Object in EPA MIB. Its value is 3 for Time Synchronization Object.

Reserved

This field is reserved.

Primary Time Server

This attribute specifies the IP address of Primary Time Server.

Secondary Time Server

This attribute specifies the IP address of Secondary Time Server.

Time Request Timeout

This attribute specifies the time (in microseconds) of time client to wait for the request from time server.

Time Request Interval

This attribute specifies the time interval (in seconds) of time client to transmit synchronization request.

Capable Time Sync Class

This attribute specifies time synchronization precision supported by time client:

0——No precision requirement;

1——Time Synchronization precision <1s;

2——Time Synchronization precision <100ms;

3——Time Synchronization precision <10ms;

4——Time Synchronization precision <1ms;

5——Time Synchronization precision <100us;

6——Time Synchronization precision <10us;

7——Time Synchronization precision <1us.

Target Time Sync Class

This attribute indicates the desired time synchronization precision supported by time client. It may be set by user application:

0——no precision requirement;

1——Time Synchronization precision <1s;

2——Time Synchronization precision <100ms;

3——Time Synchronization precision <10ms;

4——Time Synchronization precision <1ms;

5——Time Synchronization precision <100us;

6——Time Synchronization precision <10us;

7——Time Synchronization precision <1us.

Current Time

This attribute indicates current clock time in local device calculated since January 1, 1984. Its data type is TimeOfDay.

Standard Time Difference

This attribute indicates time difference between standard system time and current time. Its data type is 6 bytes TimeDifference.

8.3.2.3.3.3 Services

Read

The service allows user to read Clock Synchronization Object Class attributes.

Read service is defined in 8.3.5.3.1.

Write

The service allows user to set Clock Synchronization Object Class attributes.

Write service is defined in 8.3.5.3.2.

Clock Synchronization Service

The necessary services are defined in RFC 2030 or IEC 61588.

8.3.2.3.4 Confirmed Service Max Response Time Class

8.3.2.3.4.1 Formal model

EPA ASE:		**SYSTEM MANAGEMENT ASE**	
CLASS:		**MAX RESPONSE TIME**	
CLASS ID:		**Not Used**	
PARENT CLASS:		**TOP**	
ATTRIBUTES:			
1.	(m)	Key Attribute:	Object ID
2	(m)	Attribute:	Reserved
3.	(m)	Attribute:	Max Response Time
SERVICES:			
1.	(o)	OpsService:	Read
2.	(o)	OpsService:	Write

8.3.2.3.4.2 Attributes

Object ID

This attribute identifies Confirmed Service Max Response Time Class Object in EPA MIB. Its value is 4.

Reserved

This field is reserved.

Max Response Time

This attribute specifies max response time to wait for response since the request primitive is sent for a confirmed service. Its data type is 4 bytes of TimeDifference.

8.3.2.3.4.3 Services

Read

The service allows user to read the attributes of confirmed service max response time object.

Read service is defined in 8.3.5.3.1.

Write

The service allows user to write the attributes of confirmed service max response time object.

Write service is defined in clause 8.3.5.3.2.

8.3.2.3.5 EPA Communication Scheduling Management Class

This class specifies the information used in EPA Communication Scheduling Management entity.

8.3.2.3.5.1 Formal model

EPA ASE:		**SYSTEM MANAGEMENT ASE**	
CLASS:		**COMMUNICATION SCHEDULE MANAGEMENT**	
CLASS ID:		**Not Used**	
PARENT CLASS:		**TOP**	
ATTRIBUTES:			
1.	(m)	Key Attribute:	Object ID
2.	(m)	Attribute:	Communication Macrocycle
3.	(m)	Attribute:	NonPeriodic Data Transfer Offset
4.	(m)	Attribute:	Communication Macrocycle Version Number
SERVICES:			
1.	(o)	OpsService:	Read
2.	(o)	OpsService:	Write

8.3.2.3.5.2 Attributes

Object ID

This attribute identifies EPA Communication Scheduling Management Object in EPA MIB. Its value is 5.

Communication Macrocycle

The data type of Communication Macrocycle is 4 bytes of TimeDifference may be set by user. The default value of 0XFFFFFFFF indicates that Communication Macrocycle has not been configured.

NonPeriodic Data Transfer Offset

This attribute specifies time offset of the NonPeriodicDataTransfer starting time from the starting time of a communication macrocycle. Its data type is 4 bytes of TimeDifference. The default value of 0XFFFFFFFF indicates that this attribute has not been configured.

Communication Cycle Version Number

This attribute specifies the configuration state of EPA Communication Scheduling Management object. Each time the object is modified, the value of this attribute is incremented by 1.

8.3.2.3.5.3 Services

Read

The service allows user to read Communication Cycle Management Object Class attributes.

Read service is defined in 8.3.5.3.1.

Write

The service allows user to set Communication Cycle Management Object Class attributes.

Write service is defined in 8.3.5.3.2.

8.3.2.3.6 Device Application Information Class

8.3.2.3.6.1 Formal model

EPA ASE:	**SYSTEM MANAGEMENT ASE**
CLASS:	**DEVICE APPLICATION INFORMATION**
CLASS ID:	**Not Used**
PARENT CLASS:	**TOP**

ATTRIBUTES:

1. (m) Key Attribute: Object ID
2. (m) Attribute: XDDL Version

SERVICES:

1. (o) OpsService: Read

8.3.2.3.6.2 Attributes

Object ID

This attribute identifies Device Application Information object in EPA MIB. Its value is 6.

XDDL Version

This attribute indicates the version of XDDL document. Its date type is Unsigned16, where the higher byte specifies primary version number while the lower byte indicates the secondary version number.

8.3.2.3.6.3 Services

Read

The service allows user to read Device Application Information Object Class attributes.

Read service is defined in 8.3.5.3.1.

8.3.2.3.7 FB Application Information Header Class

8.3.2.3.7.1 Formal model

EPA ASE:		**SYSTEM MANAGEMENT ASE**	
CLASS:		**FB APPLICATION INFORMATION HEADER**	
CLASS ID:		**Not Used**	
PARENT CLASS:		**TOP**	
ATTRIBUTES:			
1.	(m)	Key Attribute:	Object ID
2.	(m)	Attribute:	Number of FB Application Information Object
3.	(m)	Attribute:	First Number of FB Application Information Object
SERVICES:			
1.	(o)	OpsService:	Read

8.3.2.3.7.2 Attributes

Object ID

This attribute identifies FB Application Information Object Header object in EPA MIB. Its value is 7.

Number of FB Application Information Object

This attribute indicates the number of FB Application Information Object in local device.

First Number of FB Application Information Object

This attribute indicates the first number of FB Application Information Object in EPA MIB. Its value shall be greater than ObjectID of domain application object header.

8.3.2.3.7.3 Services

Read

The optional service allows user to read the attributes of FB Application Information Object Header class.

Read service is defined in 8.3.5.3.1.

8.3.2.3.8 Link Object Header Class

8.3.2.3.8.1 Formal model

EPA ASE: **SYSTEM MANAGEMENT ASE**
CLASS: **LINK OBJECT HEADER**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1.	(m)	Key Attribute:	Object ID
2.	(m)	Attribute:	Number of Link Object
3.	(m)	Attribute:	First Number of Link Object
4.	(m)	Attribute:	Number of Configured Link Object
5.	(m)	Attribute:	Number of UnConfigured Link Object

SERVICES:

1.	(o)	OpsService:	Read

8.3.2.3.8.2 Attributes

Object ID

This attribute identifies Link Object Header object in EPA MIB. Its value is 8.

Number of Link Object

This attribute indicates the number of link objects in EPA MIB.

First Number of Link Object

This attribute indicates the first number of link object in EPA MIB.

Number of Configured Link Object

This attribute indicates the number of link objects configured by users.

Number of Unconfigured Link Object

This attribute indicates the number of unconfigured link objects in local device.

8.3.2.3.8.3 Service

Read

The optional service allows user to read the attributes of Link Object Header Class.

Read service is defined in 8.3.5.3.1.

8.3.2.3.9 Domain application information header class

8.3.2.3.9.1 Formal model

EPA ASE: **SYSTEM MANAGEMENT ASE**
CLASS: **DOMAIN APPLICATION INFORMATION HEADER**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1.	(m)	Key Attribute:	Object ID
2.	(m)	Attribute:	Number of Domain Application Object
3.	(m)	Attribute:	First Number of Domain Application Object
4.	(m)	Attribute:	Number of Configured Domain Object
5.	(m)	Attribute:	Number of UnConfigured Domain Object

SERVICES:

1.	(o)	Ops Service:	Read

8.3.2.3.9.2 Attributes

Object ID

This attribute identifies domain Application Information Object Header Class in EPA MIB. Its value is 9.

Number of Domain Application Information Object

This attribute indicates the number of Domain Application Information Object in local device.

First Number of Domain Application Object

This attribute indicates the first number of Domain Application Information Object in local device.

Number of Configured Domain Object

This attribute indicates the number of configured Domain objects in local device.

Number of Unonfigured Domain Object

This attribute indicates the number of unconfigured Domain objects in local device.

8.3.2.3.9.3 Services

Read

This optional service provides reading the attribute of Domain Application Object Header Class.

Read service is defined in 8.3.5.3.1.

8.3.2.3.10 FB application information class

8.3.2.3.10.1 Formal model

EPA ASE: **SYSTEM MANAGEMENT ASE**
CLASS: **FB APPLICATION INFORMATION**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1.	(m)	Key Attribute:	Object ID
2.	(m)	Attribute:	Reserved
3.	(m)	Attribute:	FB Name
4.	(m)	Attribute:	FB Type
5.	(m)	Attribute:	Max Number of Instantiation
6.	(m)	Attribute:	FB Execution Time
7.	(m)	Attribute:	First Number of Instantiation

SERVICES:

1.	(o)	Ops Service:	Read

8.3.2.3.10.2 Attributes

Object ID

This attribute identifies FB Application Information Object in EPA MIB. The number of ObjectID of FB Application Information Objects shall be appointed in series.

Reserved

This attribute is a reserved field.

FB Name

This attribute is used to identify an FB in local device.

FB Type

This attribute indicates the type of an FB.

Max Number of Instantiation

This attribute indicates the max instantiation number of an FB.

FB Execution Time

This attribute specifies the execution time (in millisecond) of an FB.

First Number of Instantiation

This attribute defines the first number which can be assigned for FB instantiation.

8.3.2.3.10.3 Services

Read

This optional service provides reading the attributes of FB Application Information Object.

Read service is defined in 8.3.5.3.1.

8.3.2.3.11 EPA link object class

8.3.2.3.11.1 Formal model

EPA ASE:		**APPLICATION RELATIONSHIP ASE**
CLASS:		**LINK OBJECT**
CLASS ID:	**Not Used**	
PARENT CLASS:	**TOP**	
ATTRIBUTES:		
1.	(m) Key Attribute:	Object ID
2.	(m) Attribute:	LocalAppID
3.	(m) Attribute:	Local Object ID
4.	(m) Attribute:	RemoteAppID
5.	(m) Attribute:	RemoteObjectID
6.	(m) Attribute:	ServiceOperation
7.	(m) Attribute:	ServiceRole
8.	(m) Attribute:	RemoteIPAddress
9.	(m) Attribute:	SendTimeOffset
SERVICES:		
1.	(o) Ops Service:	Read
2.	(o) Ops Service:	Write

8.3.2.3.11.2 Attributes

Object ID

This attribute identifies EPA Link Object in EPA MIB. The number of ObjectID of EPA Link Object shall be appointed in series.

LocalAppID

This attribute identifies local FB instance.

Local Object ID

This attribute identifies local variant object.

Remote App ID

This attribute identifies remote FB instantiation.

RemoteObjectID

This attribute identifies remote variant object.

ServiceOperation

This attribute specifies the EPA application service to be used in the relevant communication relationship:

0——local link, no EPA application service is used;

1 through 17—— the ServiceID of EPA application services is used;

Others—— invalid service.

ServiceRole

This attribute defines the AREP role of local device in communication process:

0——SENDER, indicating that the AREP role of the local device is CLIENT or PUBLISHER;

1——RECEIVER , indicating that the AREP role of the local device is SERVER or SUBSCRIBER:

Others——Link Object is invalid, and 0xFF indicates that the Link Object is not configured or the Link Object has been deleted.

RemoteIPAddress

This attribute identifies IP address of remote device. This attribute can be ignored if local FB instantiation object and remote FB instantiation object are in the same EPA device.,

SendTimeOffset

This attribute defines the time offset when the relevant message shall be sent from the start time of a communication macrocycle. This attribute is valid when Service ID is 0 (DISTRIBUTE) and ServiceRole is 0.

8.3.2.3.11.3 Service

Read

This optional service permits users to read the attributes of Link Object.

Read service is defined in clause 8.3.5.3.1.

Write

This optional service permits users to configure the attributes of Link Object.

Write Service is defined in 8.3.5.3.2.

8.3.2.3.12 Domain application information class

8.3.2.3.12.1 Formal model

EPA ASE: **SYSTEM MANAGEMENT ASE**
CLASS: **DOMAIN APPLICATION INFORMATION**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1. (m) Key Attribute: Object ID
2. (m) Attribute: Domain Object ID
3. (m) Attribute: ConfigurationStatus
4. (m) Attribute: Reserved
5. (m) Attribute: Domain Name

SERVICES:

1. (o) Ops Service: Read

8.3.2.3.12.2 Attributes

Object ID

This attribute identifies Domain Application Information Object in EPA MIB. The number of ObjectID of Domain Application Information Objects shall be appointed in series.

Domain Object ID

This attribute indicates the index of a Domain Object.

ConfigurationStatus

This attribute indicates the configuration status of a Domain Object. It is TRUE if Domain Object has been configured.

Reserved

This attribute is a reserved field.

Domain Name

This attribute identifies a local domain.

8.3.2.3.12.3 Services

Read

This optional service permits users to read the attributes of Domain Application Information Object.

Read service is defined in 8.3.5.3.1.

8.3.2.4 System management service specification

EPA System Management Entity provides the following six services shown in Table 22:

Table 22 – EPA System Management Entity Services

No.	service name	description
1	EM_FindTagQuery	This service is used to find the information of EPA device. It is transferred by unicast or broadcast.
2	EM_FindTagReply	This service is used as the reply to the FindTagQuery service.
3	EM_SetDeviceAttribute	This service is used to set the attribute of EPA device.
4	EM_GetDeviceAttribute	This service is used to read the attribute of EPA device.
5	EM_ClearDeviceAttribute	This service is use to clear its configuration data.
6	EM_DeviceAnnunciation	This service is used to periodically announce the presence of the EPA device on the network.

8.3.2.4.1 FindTagQuery service

8.3.2.4.1.1 Overview

EM_FindTagQuery is an unconfirmed service. It is used to query the IP address of an EPA device by PD_Tag to locate it on the network. This service is often sent using broadcast. A device may not receive any reply or receive several replies at the same time. The device receiving this query request primitive shall reply using EM_FindTagReply service.

An offline-configured device shall query its PD_Tag using EM_FindTagQuery service to detect the PD_Tag conflict when it is connected on the network.

8.3.2.4.1.2 Service primitives

The service parameters for EM_FindTagQuery service are shown in Table 23:

Table 23 – EM_FindTagQuery service parameters

Parameter name	.req	.ind
Argument	M	M(=)
MessageID	M	M(=)
QueryType	M	M(=)
PD_Tag	M	M(=)
FB Tag	M	M(=)
Element ID	M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

QueryType

This parameter selects the following types to query:

0——PD_Tag query

1——FB Tag query

2——ElementID query

PD_Tag

This parameter contains the Physical Device Tag.

FB Tag

This parameter contains the tag of a function block. It is required for FB tag queries.

Element ID

This parameter contains a reference to the object of a FB parameter. The FB Tag must also be present because ElementID is not unique outside of one FB.

8.3.2.4.1.3 Service procedure

The Unconfirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.2.4.2 FindTagReply service

8.3.2.4.2.1 Service overview

This service is used to send a reply to the initiator of a FindTagQuery. It returns Device ID and PD_Tag of the device queried.

8.3.2.4.2.2 Service primitives

The service parameters for EM_FindTagReply service are shown in Table 23:

Table 24 – EM_FindTagReply service parameters

Parameter name	.req	.ind
Argument	M	M(=)
MessageID	M	M(=)
Query Type	M	M(=)
Duplicate Tag Detected	M	M(=)
Queried Object IP Address	M	M(=)
Queried Object Device ID	M	M(=)
Queried Object PD_Tag	M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

QueryType

This parameter selects the type of query:

0——the following parameter contains the information by PD_Tag query;

1——the following parameter contains the information by FB Tag query;

2——the following parameter contains the information by ElementID query;

Duplicate Tag Detected

This parameter describes the duplicated status of PD_Tag among devices.

Queried Object IP Address

This parameter contains the IP address of the queried EPA device.

Queried Object Device ID

This parameter contains the ID of the queried EPA device. Its length is 32bytes.

Queried Object PD_Tag

This parameter contains the PD_Tag of the queried EPA device. Its length is 32bytes.

8.3.2.4.2.3 Service procedure

The Unconfirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.2.4.3 GetDeviceAttribute service

8.3.2.4.3.1 Service overview

EM_GetDeviceAttribute is a confirmed service. The host sends a request to get attributes of the device. After receiving EM_GetDeviceAttribute service, the device sends a Result(+) to the configuration application if performing normally, or else EPA system management entity sends a Result(-) to the configuration application.

8.3.2.4.3.2 Service primitives

The service parameters for EM_GetDeviceAttribute service are shown in Table 25:

Table 25 – EM_GetDeviceAttribute service parameters

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
Destination IP Address	M	M(=)		
Result (+)			S	S(=)
MessageID			M	M(=)
Device ID			M	M(=)
PD_Tag			M	M(=)
Status			M	M(=)
Device Type			M	M(=)
Annunciation Interval			M	M(=)
Annunciation Version Number			M	M(=)
Duplicate Tag Detected			M	M(=)
Device Redundancy Number			M	M(=)
LAN Redundancy Port			M	M(=)
Device Redundancy State			M	M(=)
Max Redundancy Number			M	M(=)
Active IP Address			M	M(=)
Result(-)			S	S(=)
MessageID			M	M(=)
Destination IP Address			M	M(=)
Error Type			M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

Destination IP Address

This parameter contains the destination IP address to which the service request is to be sent.

Result(+)

This selection type parameter indicates that the service request succeeded.

MessageID

This parameter contains the value of the MessageID in the request.

Device ID

This parameter contains the identifier of the device.

PD_Tag

This parameter contains the Physical Device Tag.

Status

This parameter contains the following status of EPA device:

0——no address

1——unconfigured

2——configured

Device Type

This parameter contains the type of the device.

Annunciation Interval

This parameter contains the interval of sending DeviceAnnunciation message.

Annunciation Version Number

This parameter contains the version number of the message annunciated.

Duplicate Tag Detected

This parameter describes the duplicated status of PD_Tag among devices.

Device Redundancy Number

This parameter contains the redundancy number of the device. Its value is 0 and the following parameters are invalid when the device is active.

LAN Redundancy Port

This conditional parameter contains the value of the Port used to receive LAN Redundancy messages. It is present in the response primitive if the value of Redundancy Number is not 0.

Device Redundancy State

This parameter contains the following redundancy status of the device:

0——active status;

1——redundancy status, it is present in the response primitive if the value of Redundancy Number is not 0.

Max Redundancy Number

This parameter contains the max redundancy number of the device. It is present in the response primitive if the value of Redundancy Number is not 0.

Active IP Address

This parameter contains the IP address of active device. It is present in the response primitive if the value of Redundancy Number is not 0.

Result(-)

This selection type parameter indicates that the service request failed.

Error Type

This parameter contains the reason that caused failure.

8.3.2.4.3.3 Service procedure

The Confirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.2.4.4 DeviceAnnunciation service

8.3.2.4.4.1 Service overview

DeviceAnnunciation is an unconfirmed service. EPA device periodically sends this service request at the rate specified by Annunciation Interval to inform the configuration application. This service is often sent by broadcast.

8.3.2.4.4.2 Service primitives

The service parameters for EM_DeviceAnnunciation service are shown in Table 26:

Table 26 – EM_DeviceAnnunciation service parameters

Parameter name	.req	.ind
Argument	M	M(=)
MessageID	M	M(=)
Device ID	M	M(=)
PD_Tag	M	M(=)
Status	M	M(=)
Device Type	M	M(=)
Annunciation Version Number	M	M(=)
Device Redundancy Number	M	M(=)
Device Redundancy State	M	M(=)
LAN Redundancy Port	M	M(=)
Duplicate Tag Detected	M	M(=)
Max Redundancy Number	M	M(=)
Active IP Address	M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

Device ID

This parameter contains the identifier of the device.

PD_Tag

This parameter contains the Physical Device Tag.

Status

This parameter contains the following status of EPA device:

0——no address

1——unconfigured

2——configured

Device Type

This parameter contains the type of the device. It is used to describe the functions of the device and defined by the manufacturer.

Annunciation Version Number

This parameter contains the version number of the message annunciated.

Device Redundancy Number

This parameter contains the redundancy number of the device.

LAN Redundancy Port

This conditional parameter contains the value of the Port used to receive LAN Redundancy messages.

Duplicate Tag Detected

This parameter describes the duplicated status of PD_Tag among devices.

Max Redundancy Number

This parameter contains the max redundancy number of the device.

Active IP Address

This parameter contains the IP address of active device.

8.3.2.4.4.3 Service procedure

The Unconfirmed Service Procedure specified in 8.1.6 applies to this service.

8.3.2.4.5 SetDeviceAttribute service

8.3.2.4.5.1 Service overview

EM_SetDeviceAttribute is a confirmed service which is sent using unicast. User application sends this service request to set the PD_Tag and others attributes of the device.

In order to avoid error, the DeviceID parameter within the request must be equal to the DeviceID of the device. When executing, if the device has already has a PD_Tag, it must clear its PD_Tag at first using ClearDeviceAttribute service.

8.3.2.4.5.2 Service primitives

The service parameters for EM_SetDeviceAttribute service are shown in Table 27:

Table 27 – EM_SetDeviceAttribute service primitives

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
Destination IP Address	M	M(=)		
Device ID	M	M(=)		
PD_Tag	M	M(=)		
Annunciation Interval	M	M(=)		
Duplicate Tag Detected	M	M(=)		
Device Redundancy Number	M	M(=)		
LAN Redundancy Port	M	M(=)		
Device Redundancy State	M	M(=)		
Max Redundancy Number	M	M(=)		
Active IP Address	M	M(=)		
Result (+)			S	S(=)
MessageID			M	M(=)
Destination IP Address			M	M(=)
Max Redundancy Number			M	M(=)
Result (-)			S	S(=)
MessageID			M	M(=)
Destination IP Address			M	M(=)
Error Type			M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

Destination IP Address

This parameter is the IP address to which the service request is to be sent.

Device ID

This parameter contains the id of the device. Its length is 32bytes.

PD_Tag

This parameter contains the Physical Device Tag. Its length is 32bytes.

Annunciation Interval

This parameter contains the interval of sending annunciation message. Its unit is second.

Duplicate Tag Detected

This parameter describes the duplicated status of PD_Tag among devices.

Device Redundancy Number

This parameter contains the redundancy number of the device. Its value is 0 and the following parameters are invalid when the device is active.

LAN Redundancy Port

This conditional parameter contains the value of the Port used to receive LAN Redundancy messages.

Device Redundancy State

This parameter contains the following redundancy status of the device:

0——active status;

1——redundancy status, it is present in the response primitive if the value of Redundancy Number is not 0.

Max Redundancy Number

This parameter contains the max redundancy number of the device. It is present in the response primitive if the value of Redundancy Number is not 0.

Active IP Address

This parameter contains the IP address of active device. It is present in the response primitive if the value of Redundancy Number is not 0.

Result(+)

This selection type parameter indicates that the service request succeeded.

MessageID

This parameter contains the value of the MessageID in the request.

Max Redundancy Number

This parameter contains the max redundancy number of the device. It is present in the response primitive if the value of Redundancy Number is not 0.

Result(-)

This selection type parameter indicates that the service request failed.

Error Type

This parameter contains the reason that caused failure.

8.3.2.4.5.3 Service procedure

The Confirmed Service Procedure specified in 8.1.6 applies to this service.

8.3.2.4.6 ClearDeviceAttribute service

8.3.2.4.6.1 Service overview

EM_ClearDeviceAttribute is a confirmed service which is sent by unicast. User application sends this service request to clear the PD_Tag and set attributes of the device to default value. In order to avoid error, the parameter (DeviceID and PD_Tag) within the request must be equal to the DeviceID and PD_Tag of the device.

8.3.2.4.6.2 Service primitives

The service parameters for EM_SetDeviceAttribute service are shown in Table 28:

Table 28 – EM_ClearDeviceAttribute service parameter

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
Destination IP Address	M(=)	M(=)		
Device ID	M	M(=)		
PD_Tag	M	M(=)		
Result(+)			S	S(=)
MessageID			M	M(=)
Destination IP Address			M	M(=)
Result(-)			S	S(=)
MessageID			M	M(=)
Destination IP Address			M	M(=)
Error Type			M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

Destination IP Address

This parameter is the IP address to which the service request is to be sent.

Device ID

This parameter contains the id of the device. Its length is 32 bytes.

PD_Tag

This parameter contains the Physical Device Tag. Its length is 32 bytes.

Result(+)

This selection type parameter indicates that the service request succeeded.

MessageID

This parameter contains the value of the MessageID in the request.

Result(-)

This selection type parameter indicates that the service request failed.

Error Type

This parameter contains the reason that caused failure.

8.3.2.4.6.3 Service procedure

The Confirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.3 Domain ASE

8.3.3.1 Overview

A domain represents a memory area whose contents may be data or program, and the memory area is represented as an ordered sequence of octets. A domain can be uploaded or downloaded, to maintain integrity, only one download or upload operation for a domain is permitted at the same time.

The domain ASE provides following services shown in Table 29:

Table 29 – Services for Domain ASE

Service name	Service	Type	Object
DomainDownload	Download service for Domain	Confirmed	Domain Object
DomainUpload	Upload service for Domain	Confirmed	

8.3.3.2 Domain model specification

8.3.3.2.1 Formal model

EPA ASE: **DOMAIN ASE**
CLASS: **DOMAIN**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1. (m) Key Attribute: Object ID
2. (m) Key Attribute: Domain Name
3. (m) Attribute: Max Octets
4. (m) Attribute: Password
5. (m) Attribute: Access Groups
6. (m) Attribute: Access Rights
7. (m) Attribute: Local Address
8. (m) Attribute: Domain State
9. (m) Attribute: Last State
10. (m) Attribute: Used Application Counter

SERVICES:

1. (o) Ops Service: DomainDownload
2. (o) Ops Service: DomainUpload

8.3.3.2.2 Attributes

Object ID

This attribute is the object identifier.

Domain Name

This attribute contains the name of domain object.

Max Octets

This attribute specifies the maximum size of domain in octets.

Password

This attribute contains the password for the access right.

Access Groups

This attribute allows restricting the access only to clients who belong to one of the groups defined by the attribute.

This attribute specifies if the object belongs to a group of users, as shown in Table 30. If one of the bits is set to 1, it indicates a group number to which the object belongs.

Table 30 – Access Groups for Domain

Bit	Signification
7	Access Group 1
6	Access Group 2
5	Access Group 3
4	Access Group 4
3	Access Group 5
2	Access Group 6
1	Access Group 7
0	Access Group 8

Access Rights

This attribute defines the type of access defined for domain, as show in Table 31. The corresponding access right is permitted while related bit has been set.

Table 31 – Access Rights for Domain

Bit	Name	Signification
7	R	Right to Read for the registered Password
6	W	Right to Write for the registered Password
5	U	Right to Use the registered Password
3	Rg	Right to Read for the Access Groups
2	Wg	Right to Write for the Access Groups
1	Ug	Right to Use the Access Groups

Local Address

This attribute is a locally significant address of domain. The value 0xFFFFFFFF indicates that no local address is available.

Domain State

This attribute specifies the following states of domain:

0——EXISTENT

1——DOWNLOADING

2——UPLOADING

3——READY

4——IN-USE

Last State

This attribute specifies the state of domain before uploading or downloading.

Used Application Counter

This attribute contains the number of the program using this domain. If the value of this attribute is more than 0, it is indicated that the domain is being used. The domain can not be re-downloaded while it is being used.

8.3.3.2.3 Services

DomainDownload

This optional service permits the client to download a domain.

DomainUpload

This optional service permits the client to upload a domain.

8.3.3.3 Domain ASE service specification

8.3.3.3.1 Domain Download service

8.3.3.3.1.1 Service overview

Data and program can be downloaded by Domain Download service.

The following is the service procedure for Domain Downloading:

Step 1:

The user application examines at first whether the length of downloading data exceeds the max length (512 bytes) or not. It shall divide the data into several packages numbered orderly if the length of downloading data exceeds the max length.

Step 2:

User transmits the numbered data packet and the corresponding parameters by calling DomainDownload service.

Step 3:

After coding and packing, EPA Domain ASE transfers service package to EPA Socket Mapping ASE and sends it on the network by invocating UDP service.

Step 4:

After receiving package from UDP port, EPA device (the sever) decodes it and store the downloading data into local memory. Then, it returns a Result(+) response.

Step 5:

When receiving response, user application will operate as follows:

a) if receiving a Result(+) which indicates that the last downloading operation is successful. User application will repeat downloading operation using step 2 through step 5 till all Second if there remains other data to be downloaded. Otherwise, it ends the downloading operation.

b) if receiving a Result(-), It will report the error to users.

8.3.3.3.1.2 Service primitives

The service parameters for Domain Download service are shown in Table 32:

Table 32 – Parameters for Domain Download Service

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
SourceAppID	M	M(=)		
DestinationAppID	M	M(=)		
DestinationObject ID	M	M(=)		
DataNumber	M	M(=)		
MoreFollows	M	M(=)		
DataLength	M	M(=)		
LoadData	M	M(=)		
Result(+)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
Result(-)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
ErrorType			M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

SourceAppID

This parameter contains the identifier of the source FB instantiation.

DestinationAppID

This parameter contains the identifier of the destination FB instantiation.

Destination Object ID

This parameter contains the identifier of the domain object.

DataNumber

This parameter contains the number of downloading data packet. It is numbered from 0.

MoreFollows

This parameter indicates whether there remains other data or not. When there remains other data to be downloaded, it is set to TRUE.

DataLength

This parameter contains the length of the downloading data.

LoadData

This parameter contains the downloading data.

Result (+)

This selection type parameter indicates that the service request succeeded.

MessageID

This parameter contains the value of the MessageID in the request.

DestinationAppID

This parameter contains the identifier of the destination FB instantiation.

Result (-)

This selection type parameter indicates that the service request failed.

DestinationAppID

This parameter contains the identifier of the destination FB instantiation.

Error Type

This parameter contains the reason that caused failure.

8.3.3.3.1.3 Service procedure

The Confirmed Service Procedure specified in 8.1.6 applies to this service.

8.3.3.3.2 Domain Upload service

8.3.3.3.2.1 Service overview

Data and program can be uploaded by Domain Upload service.

The following is the service procedure for Domain Uploading:

Step 1:

User application (client) transfers the uploading request with uploading domain object to EPA device (server) using DomainUpload service.

Step 2:

When receiving Domain Upload request from UDP port, the server will examine if the domain can be uploaded. If yes, it will compare the length of uploading data exceeds with the max length (512 bytes) of uploading data in a time. It will divide the uploading data into several packages numbered orderly if the length of uploading data exceeds with the max length.

Step3 :

The server transmits the numbered data packet and the corresponding parameters as response parameters.

Step 4:

When receiving response, user shall operate as follows:

a) if receiving a Result(+) response, it shall send Domain Upload request if the MoreFollows parameter in the response is TRUE. Otherwise, it ends Domain Uploading operation.

b) if receiving a Result(-) response, if report the error to users.

8.3.3.3.2.2 Service primitives

The service parameters for Domain Upload service are shown in Table 33:

Table 33 – Parameters for Domain Upload Service

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
SourceAppID	M	M(=)		
DestinationAppID	M	M(=)		
DestinationObject	M	M(=)		
ID	M	M(=)		
DataNumber				
			S	S(=)
Result(+)			M	M(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
MoreFollows			M	M(=)
DataLength			M	M(=)
LoadData				
			S	S(=)
Result(-)			M	M(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
ErrorType				

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by adding 1.

SourceAppID

This parameter contains the identifier of the source FB instantiation.

DestinationAppID

This parameter contains the identifier of the destination FB instantiation.

Destination Object ID

This parameter contains the identifier of the domain object.

DataNumber

This parameter contains the number of service invocated during uploading. It is numbered from 0.

Result(+)

This selection type parameter indicates that the service request succeeded.

MoreFollows

This parameter indicates whether there remains other data or not. When there remains other data to be uploaded, it is set to TRUE.

DataLength

This parameter contains the length of the uploading data.

LoadData

This parameter contains the uploading data..

Result(-)

This selection type parameter indicates that the service request failed.

Error Type

This parameter contains the reason that caused failure.

8.3.3.3.2.3 Service procedure

The Confirmed Service Procedure specified in 8.1.6 applies to this service.

8.3.4 Event ASE

8.3.4.1 Overview

The event ASE is mainly used to send important messageS from one to other devices (or in broadcast mode, to all other devices). It is the user who defines the conditions that triggers the event. The application process invokes the EventNotification service to notify the event when the conditions are met. It's also the user who confirms the event.

Through the Event Notification service, the user transmits the Event Number and Event Data, e.g. measurement value, states and unit. The management of the Event Number is completed by the user. The Event Notificaiton may represent a collective alarm, where the data may contain information about which channel has trigged the collective alarm.

The receiver of the Event Notification may acknowledge the Event by using the AcknowledgeEventNotification service. The counter value is transmitted with the AcknowledgeEventNotification service. The counter serves to correlate the EventNotification and the AcknowledgeEventNotification.

The receiver of the EventNotification may lock or unlock the EventNotification using the AlterEventConditionMonitoring service.

The EPA event ASE provides three services of event management, as shown in Table 34.

Table 34 – Service for event ASE

Service name	Service	Confirmed/ Unconfirmed	Object
Event Notification	Event Notification	Unconfirmed	Event Object
AcknowledgeEventNotification	Event Acknowledge	Confirmed	
Alter Event Condition Monitor	Alter Event Condition Monitor	Confirmed	

8.3.4.2 Event model class specification

8.3.4.2.1 Formal model

EPA ASE: **EVENT ASE**
CLASS: **EVENT**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1. (m) key attribute: Object ID
2. (m) attribute: Length
3. (m) attribute: Password
4. (m) attribute: Access Groups
5. (m) attribute: Access Rights
6. (m) attribute: Local Address
7. (m) attribute: Enabled

SERVICES:

1. (o) Ops Service: EventNotification
2. (o) Ops Service: AcknowledgeEventNotification
3. (o) Ops Service: AlterEventConditionMonitor

8.3.4.2.2 Attributes

Object ID

This attribute indicates the identifier of the event object.

Length

This attribute indicates the max data length of event.

Password

This attribute contains the password for the access rights. Its value is null if it is not used.

Access Groups

This attribute allows restricting the access only to clients who belong to one of the groups defined by the attribute.

This attribute specifies if the object belongs to a group of users, as shown in Table 35. If one of the bits is set to 1, it indicates a group number to which the object belongs.

Table 35 – Access group attribute detail for event object

Bit name	Number of access group
7	Access group 1
6	Access group 2
5	Access group 3
4	Access group 4
3	Access group 5
2	Access group 6
1	Access group 7
0	Access group 8

Access Rights

This attributes contains the access right associated with the Event object, as shown in Table 36. Each bit positioned at 1 indicates the acceptance conditions of corresponding access right.

Table 36 – Access rights attribute details for event object

Bit	Name	Signification
7	R	Right to Read for the registered Password
6	W	Right to Write for the registered Password
5	U	Right to Use the registered Password
3	Rg	Right to Read for the Access Groups
2	Wg	Right to Write for the Access Groups
1	Ug	Right to Use the Access Groups

Local Address

The pointer of the specific object, it may be used internally for addressing the object. If it's not need, the value should be set as 0xFFFFFFFF.

Enabled

This attribute indicates the state of event object that is defined by State-Machine.

Enabled = TRUE ⇔ UNLOCKED

Indicates event object is unlocked and can be sent.

Enabled = TRUE ⇔ LOCKED

Indicates event object is locked and can't be sent.

8.3.4.2.3 Services

EventNotification

This optional service allows the server to notify one or more event.

AcknowledgeEventNotification

This optional service enables a Client to acknowledge several event occurrences.

AlterEventConditionMonitor

This optional service enables user to lock or unlock a event object.

8.3.4.3 Event ASE Service specification

8.3.4.3.1 EventNotification service

8.3.4.3.1.1 Service overview

This service is used to transmit event notification. The event is transmitted by calling EventNotification Service. It is an unconfirmed service using multicast or broadcast mode.

8.3.4.3.1.2 Service Primitive

The service parameters for this service are shown in Table 37.

Table 37 – EventNotification Service Parameters

Parameter name	.req	.ind
Argument	M	M(=)
MessageID	M	M(=)
DestinationAppID	M	M (=)
SourceAppID	M	M (=)
SourceObjectID	M	M (=)
EventNumber	M	M (=)
EventData	U	U (=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by 1.

DestinationAppID

This parameter is used to indicate the identifier of the destination application

SourceAppID

This parameter contains the identifier of the source FB instantiation.

SourceObjectID

This parameter indicates the identifier of the object associated with this event.

EventNumber

The parameter indicates number of the event.

EventData

This parameter includes the specific data. The data of object should be mapped to the data of parameter according to the description of the data type.

8.3.4.3.1.3 Service procedure

The Unconfirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.4.3.2 AcknowledgeEventNotification service

8.3.4.3.2.1 Service Overview

The service allows the user to acknowledge the Event Notification. It is a confirmed service. After the Event Notification is received, the user invokes this service to acknowledge the event.

8.3.4.3.2.2 Service Primitive

The service parameters for this service are shown in Table 38.

Table 38 – AcknowledgeEventNotification Service Parameters

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
DestinationAppID	M	M(=)		
DestinationObjectID	M	M(=)		
EventNumber	M	M(=)		
Result(+)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
Result(-)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
ErrorType			M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by 1.

DestinationAppID

This parameter is used to indicate the identifier of the destination application

DestinationObjectID

This parameter indicates the identifier of the object associated with this event.

EventNumber

The parameter indicates number of the event

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Error Type

This parameter contains the reason that caused failure.

8.3.4.3.2.3 Service procedure

The Unconfirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.4.3.3 AlterEventConditionMonitor Service

8.3.4.3.3.1 Service Overview

AlterEventConditionMonitor Service permits locking or unlocking the event object. The service is the confirmed service.

8.3.4.3.3.2 Service Primitive

The service parameters for this service are shown in Table 39.

Table 39 – AlterEventConditionMonitor Service Parameters

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
DestinationAppID	M	M(=)		
DestinationObjectID	M	M(=)		
Enabled	M	M(=)		
Result(+)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
Result(-)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
Error Type			M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by 1.

DestinationAppID

This parameter is used to indicate the identifier of the destination application

ObjectID

This parameter indicates the identifier of the object associated with this event.

Enabled

The boolean Parameter, is used to set the value of the corresponding attribute of the event object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

ErrorType

This parameter contains the reason that caused failure.

8.3.4.3.3.3 Service procedure

The Unconfirmed Service Procedure specified in 8.1.6 applies to this service.

8.3.5 Variable ASE

8.3.5.1 Overview

The Variable ASE defines the network visible attributes of application data and provides a set of services to read, write and distribute their values.

Variable ASE supports the following three types of variable object:

a) Simple Variable Object

The Simple Variable Object represents a single, simple variable that is characterized by a defined Data Type.

b) Array Variable Object

The Array Object is used to define a constructed variable in which all elements have the same Data Type and length. The Array may be accessed completely.

c) Structure Variable Object

The Structure Object defines the variable that consists of a collection of Simple Variable with different Data Type. The Structure Object may be accessed completely or element-wise.

The typical structure object includes FB and FB parameter.

Each Variable Object is identified through an unique Object ID for its logical location. The Object ID is used for manipulating the Variable Object.

The services supported by variable ASE are shown in Table 40:

Table 40 – Variable Access Services

Service Name	Service	Confirmed/Unconfirmed	Object
Read	Read	Confirmed	Single Variable Array Variable Structure Variable
Write	Write	Confirmed	
Distribute	Distribute	Unconfirmed	

8.3.5.2 Variable model class Specification

8.3.5.2.1 Simple variable class Specification

8.3.5.2.1.1 Formal model

EPA ASE: **VARIABLE ASE**
CLASS: **SIMPLE VARIABLE**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**

ATTRIBUTES:

1.	(m)	key attribute:	Object ID
2.	(m)	attribute:	Data Type
3.	(m)	attribute:	Length
4.	(m)	attribute:	Local Address
5.	(m)	attribute:	Password
6.	(m)	attribute:	Access Groups
7.	(m)	attribute:	Access Rights

SERVICES:

1.	(o)	Ops Service:	Read
2.	(o)	Ops Service:	Write
3.	(o)	Ops Service:	Distribute

8.3.5.2.1.2 Attributes

Object ID

This attribute indicates the identifier of variable object.

Data Type

This attribute indicates the type of variable data.

Length

This attribute indicates the length of variable data.

Local Address

This attribute is a locally significant address of variable object. The value of 0xFFFFFFFF indicates that no local address is available.

Password

This attribute contains the password for the access right.

Access Groups

This attribute allows restricting the access only to clients who belong to one of the groups defined by the attribute.

This attribute specifies if the object belongs to a group of users, as shown in Table 41. If one of the bits is set to 1, it indicates a group number to which the object belongs.

Table 41 – Access group attribute detail for Simple Variable

bit	Number of access group
7	Access group 1
6	Access group 2
5	Access group 3
4	Access group 4
3	Access group 5
2	Access group 6
1	Access group 7
0	Access group 8

Access Rights

This attribute defines the type of access defined for domain in Table 42. The corresponding access right is permitted while related bit has been set.

Table 42 – Access rights attribute details for Simple Variable

Bit	Name	Signification
7	R	Right to Read for the registered Password
6	W	Right to Write for the registered Password
5	U	Right to Use the registered Password
3	Rg	Right to Read for the Access Groups
2	Wg	Right to Write for the Access Groups
1	Ug	Right to Use the Access Groups

8.3.5.2.1.3 Services

Read

This optional service may be used to read value of variable object.

Write

This optional service may be used to update value of variable object.

Distribute

This optional service may be used to distribute value of variable.

8.3.5.2.2 Array Variable Class specification

8.3.5.2.2.1 Formal model

EPA ASE: **VARIABLE ASE**
CLASS: **ARRAY VARIABLE**
CLASS ID: **Not Used**
PARENT CLASS: **SIMPLE VARIABLE**
ATTRIBUTES:

1. (m) key attribute: Object ID
2. (m) attribute: Number of Elements

SERVICES:

1. (o) Ops Service: Read
2. (o) Ops Service: Write
3. (o) Ops Service: Distribute

8.3.5.2.2.2 Attributes

Object ID

This attribute indicates the identifier of variable object.

Number of Elements

This attribute specifies the number of array elements for variable objects that are defined as array.

8.3.5.2.2.3 Services

Read

This optional service may be used to read values of variable object.

Write

This optional service may be used to update values of variable object.

Distribute

This optional service may be used to distribute value of variable.

8.3.5.2.3 Structure Variable Class specification

8.3.5.2.3.1 Formal model

EPA ASE: **VARIABLE ASE**
CLASS: **STRUCTURE VARIABLE**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1. (m) key attribute: Object ID
2. (m) attribute: Data Type
3. (m) attribute: Length
4. (m) attribute: Local Address
5. (m) attribute: Password
6. (m) attribute: Access Groups
7. (m) attribute: Access Rights

SERVICES:

1. (o) Ops Service: Read
2. (o) Ops Service: Write
3. (o) Ops Service: Distribute

8.3.5.2.3.2 Attributes

Object ID

This attribute indicates the identifier of variable object.

Data Type

This attribute specifies the data type of variable.

Length

This attribute is the length of variable data.

Local Address

This attribute is a locally significant address of variable object. The value of 0xFFFFFFFF indicates that no local address is available.

Password

This attribute contains the password for the access right.

Access Groups

This attribute allows restricting the access only to clients who belong to one of the groups defined by the attribute.

This attribute specifies if the object belongs to a group of users, as shown in Table 43. If one of the bits is set to 1, it indicates a group number to which the object belongs.

Table 43 – Access group attribute for Structure Variable Object

bit	Number of access group
7	Access group 1
6	Access group 2
5	Access group 3
4	Access group 4
3	Access group 5
2	Access group 6
1	Access group 7
0	Access group 8

Access Rights

This attribute defines the type of access defined for variable, as shown in Table 44. The corresponding access right is permitted while related bit has been set.

Table 44 – Access rights attribute for Structure Variable Object

Bit	Name	Signification
7	R	Right to Read for the registered Password
6	W	Right to Write for the registered Password
5	U	Right to Use the registered Password
3	Rg	Right to Read for the Access Groups
2	Wg	Right to Write for the Access Groups
1	Ug	Right to Use the Access Groups

8.3.5.2.3.3 Services

Read

This optional service may be used to read value of variable object.

Write

This optional service may be used to update value of variable object.

Distribute

This optional service may be used to distribute value of variable object.

8.3.5.3 Variable ASE service specification

8.3.5.3.1 Read service

8.3.5.3.1.1 Service overview

This confirmed service may be used to read the value of a variable object. The value is transform between devices through peer-to-peer.

8.3.5.3.1.2 Service Primitive

The service parameters for this service are shown in Table 45.

Table 45 – Read service parameters

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
DestinationAppID	M	M(=)		
DestinationObjectID	M	M(=)		
SubIndex	M	M(=)		
Result(+)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
MessageID			M	M(=)
DestinationApptID			M	M(=)
Error Type			M	M(=)

Argument

This parameter carries the parameters of the service invocation.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by 1.

DestinationAppID

This parameter contains the identifier of the destination FB instantiation.

DestinationObjectID

This parameter indicates the identifier of the object associated with read object. The DestinationObjectID is 0 indicate that read all parameters of this FB instantiation.

SubIndex

This parameter indicates the sub-index of variable object. The value of 0 indicates that all element of the object shall be read.

Result(+)

This selection type parameter indicates that the service request succeeded.

Data

This parameter contains the value to read. It must accord with the data type of the read object.

Result(-)

This selection type parameter indicates that the service request failed.

ErrorType

This parameter contains the reason that caused failure.

8.3.5.3.1.3 Service procedure

The Confirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.5.3.2 Write Service

8.3.5.3.2.1 Service Overview

This confirmed service is used to write the value of a variable object. The value is transform between devices through peer-to-peer.

8.3.5.3.2.2 Service Primitive

The service parameters for this service are shown in Table 46.

Table 46 – Write Service parameters

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
MessageID	M	M(=)		
DestinationAppID	M	M(=)		
DestinationObjectID	M	M(=)		
SubIndex	M	M(=)		
Data	M	M(=)		
Result(+)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
Result(-)			S	S(=)
MessageID			M	M(=)
DestinationAppID			M	M(=)
ErrorType			M	M(=)

Argument

The argument contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by 1.

DestinationAppID

This parameter contains the identifier of the destination FB instantiation.

DestinationObjectID

This parameter indicates the identifier of the variable object. The value of is 0 indicates that all parameters of this FB instantiation shall be written.

SubIndex

This parameter indicates the sub-index of read object. If the SubIndex is 0 indicate that all element of the object be read.

Result(+)

This selection type parameter indicates that the service request succeeded.

Data

This parameter contains the value to write.

Result(-)

This selection type parameter indicates that the service request failed.

ErrorType

This parameter contains the reason that caused failure.

8.3.5.3.2.3 Service procedure

The confirmed Service Procedure specified in clause 8.1.6 applies to this service.

8.3.5.3.3 Distribute Service

8.3.5.3.3.1 Service Overview

This service is used to transmit the specific value of Simple Variable, Array Variable and Record Variable. The main function of this service is to transmit input/output parameters among function blocks in the field devices, and it is an unconfirmed service. This service could be mapped into multicast for transmitting data to many devices.

8.3.5.3.3.2 Service Primitive

The service parameters for this service are shown in Table 47.

Table 47 – Distribute Service parameters

Parameter Name	.req	.ind
Argument	M	M(=)
MessageID	M	M(=)
Source AppID	M	M(=)
Source Object ID	M	M(=)
Data	M	M(=)

Argument

This parameter contains the parameters of the service request.

MessageID

This parameter contains the invocated number of the service. Each time this service is invocated, the value of this parameter is incremented by 1.

Source AppID

This parameter contains the identifier of the source FB instantiation.

Source Object ID

This parameter indicates the identifier of the object associated with distribute object.

Data

This parameter contains the values to be distributed.

8.3.5.3.3.3 Service procedure

The Unconfirmed Service Procedure specified in 8.1.6 applies to this service.

8.3.6 Application relationship ASE

8.3.6.1 Overview

8.3.6.1.1 General

In a distributed system, application processes communicate with each other so as to exchange application layer messages across well-defined application layer communications channels. These communication channels are abstracted in the AL as application relationships (ARs).

ARs are responsible for conveying messages between applications according to specific communication characteristics required by time-critical systems. Different combinations of these characteristics lead to the definition of different types of ARs. The characteristics of ARs are defined formally as attributes of AR Endpoint classes.

An implicit AR is defined at application layer, but explicit AR ASE entity is not defined. This implicit AR is used to establish all remote context relationship management and release communication channel.

8.3.6.1.2 AR Endpoint

8.3.6.1.2.1 Overview

Each AP involved in an AR contains an endpoint of the AR. Each AR endpoint is defined within the AE of the AP. Each endpoint definition contains a set of compatibility-related characteristics. These characteristics need to be configured appropriately for each endpoint for the AR to operate properly. The endpoint context is used by the AR ASE to manage the operation of the endpoint and the delivery of APDUs.

The messages that are conveyed by ARs are EPA application layer service requests and responses. Each of these messages is submitted to the AR ASE for transfer by an AL ASE. Figure 17 illustrates this concept.

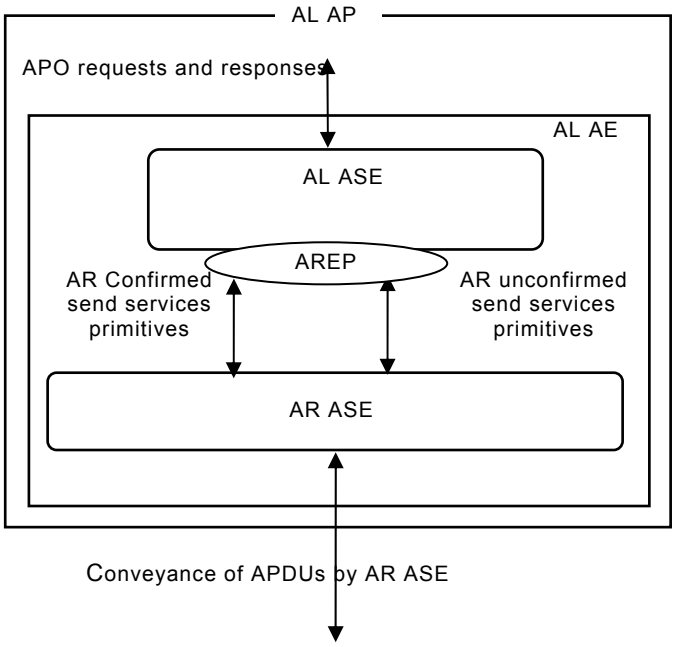


Figure 17 – The AR ASE conveys APDUs between AP

8.3.6.1.2.2 AR Endpoint role

The role of an AREP determines the permissible behavior of an AP at the AREP. An AREP role may be that of a client, server, peer (client and/or server), push publisher, push subscriber.

8.3.6.1.2.3 AR cardinality

From the point of view of a client or publisher endpoint, AR cardinality determines how many remote application processes are included in an AR. Cardinality is never expressed from the viewpoint of a server or a subscriber.

From the viewpoint of a client or peer endpoint, ARs are always 1-to-1. Client is never capable of issuing a request and waiting for responses from multiple servers.

From the viewpoint of a subscriber, ARs are 1-to-multiple. An AREP communicate with multiple or all device's AREP.

8.3.6.1.2.4 Connection model

EPA AREP uses connectionless mode based on UDP/IP. This mode neither establishes nor releases connection. It is impossible to supervise connectionless AR.

8.3.6.1.2.5 Transfer Type

EPA AREP support periodic or Non-periodic data transferring modes.

a) Periodic data transferring

In this mode, application relationship is established after application process initiates a request. After that, it performs data transferring periodically.

b) Non-Periodic data transferring

In this mode, only one data transferring is performed each time application process request is sent.

c) Both

This mode supports both periodic and non-periodic mode.

8.3.6.1.2.6 Transfer Feature

EPA AREP transfer data using either buffer or queue service.

8.3.6.1.2.7 Service Type

EPA AREP support either confirmed or unconfirmed service.

8.3.6.1.2.8 Service Relationship

EPA AREP supports both Client/Service and Publisher/Subscriber relationships.

8.3.6.1.3 AR establishment

EPA AREP uses locally established UDP channel to transfer APDU. It is established implicitly without any other services.

8.3.6.2 Application relationship endpoint class specification

8.3.6.2.1 Formal model

This model defines the common characteristics for all implicit AR endpoints. This implicit AR is not permitted to be written.

EPA ASE: **AR ASE**
CLASS: **AR ENDPOINT**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
MANAGEMENT ATTRIBUTES:

1.	(m)	key attribute:	Object ID
2.	(m)	attribute:	Local AP
3.	(m)	attribute:	Role
4.	(m)	attribute:	Initiator
5.	(m)	attribute:	TransferType
6.	(m)	attribute:	Service Type
7.	(m)	attribute:	Transfer Feature

8.3.6.2.2 Attributes

Object ID

This attribute indicates the identifier of the communication point.

Local AP

This attribute identifies the AP attached or configured to use the APEP using a local reference.

Role

This attribute specifies the role of AREP. Its value is shown as follows:

PEER -- It indicates the role of AREP is Client, Service, or Client and Service at the same time. If the role of local AREP is both Client and Server, it is specified that if local AREP is the initiator.

CLIENT – It indicates the role of AREP is client,

SERVER – It indicates the role of AREP is server.

PUBLISHER – It indicates the role of AREP is publisher.

SUBSCRIBER – It indicates the role of AREP is subscriber.

Initiator

This attribute indicates if the AREP is an communication initiator.

If Role is PEER, it is disabled.

If Role is Client or Publisher, it is set to TRUE.

If Role is Server or Subscriber, it is set to FALSE.

TransferType

This attribute indicates the transfer mode that the AREP supported.

PERIODIC – indicates the AREP support periodic data transfer service

NON PERIODIC –indicates the AREP support non-periodic data transfer service

BOTH -- indicates the AREP support both periodic and non-periodic data transfer service

ServiceType

This attribute indicates the service type that the AREP supported.

CONFIRMED – indicates that the AREP support confirmed data transfer service.

UN CONFIRMED –indicates that the AREP support unconfirmed data transfer service.

BOTH -- indicates that the AREP support both confirmed and unconfirmed data transfer service.

Transfer Feature

This attributes indicates the transferring type that the AREP supported.

QUEUE -- indicates the AREP support queued service transfer data.

BUFFER – indicates the AREP support buffer service transfer data.

8.3.6.2.3 Services

No service is defined.

8.3.7 EPA Socket Mapping ASE

8.3.7.1 Overview

EPA socket mapping ASE (ESM ASE) is application service element of EPA Socket Mapping Entity. It provides the mapping between the application services and lower layer. Its main functions are as shown follows:

a) Mapping the application layer PDU to UDP protocol;

b) Delivering the Application Layer PDUs into corresponding unsent buffer or queue according to the ServiceID identifier encapsulated in the PDUs.

c) Providing overtime diagnosis and control for the confirmed services and returning positive or negative response to the corresponding entities;

d) Providing priority management of application layer services;

e) Providing monitoring of the status of EPA Link and reporting it to users.

8.3.7.1.1 EPA Message transmitting management procedure

When receiving PDUs from EPA application access entity and EPA system management entity, EPA socket mapping entity firstly pushes it into corresponding queues according to the transmitting priority.

For the confirmed service PDUs, EPA socket mapping entity will create a timer object it according to its ServiceID and MessageID. It will start the timer when the PDU is sent to network by the ECSME and maintain it as follows:

a) If the positive response is received before the Max Response Time of confirmed message, EPA socket mapping entity delivers Result (+) response with relevant data to corresponding EPA application layer entity, and then deletes the timer object.

b) If the negative response is received before the Max Response Time of confirmed message, EPA socket mapping entity delivers Result (-) response with error code to corresponding EPA application layer entity, and then deletes the timer object.

c) If no response is received when the timer runs over, EPA socket mapping entity delivers a negative response with Error class of excess time to corresponding EPA application layer entity, and then deletes the timer object.

8.3.7.1.2 EPA Message receiving management procedure

After powered-on and initialization, the EPA device starts monitoring its communication port. When an EPA message is received, local EPA socket mapping entity will deal with it according to the ServiceID encapsulated in it as shown in Figure 18:

a) if the message received from the EPA System Management port, the EPA socket mapping entity will deal with it according to the Service ID. If the Service ID is invalid, the message will be discarded. Otherwise:

   1) If the message is a request from remote devices, it will be delivered to EPA system management entity. If it is a confirmed service request, local EPA system management entity will reply a response.

   2) If the message is the response for the local request, EPA socket mapping entity will deliver it to local EPA system management entity and delete the corresponding timer object.

b) if the message received from the EPA Application Access port, the EPA socket mapping entity will deal with it according to the Service ID. If the Service ID is invalid, the message will be discarded. Otherwise:

   1) If the message is a request from remote devices, it will be delivered to EPA application access entity. If it is a confirmed service request, local EPA application access entity will reply a response.

2) If the message is the response for the local request, EPA socket mapping entity will deliver it to local EPA application access entity and delete the corresponding timer object.

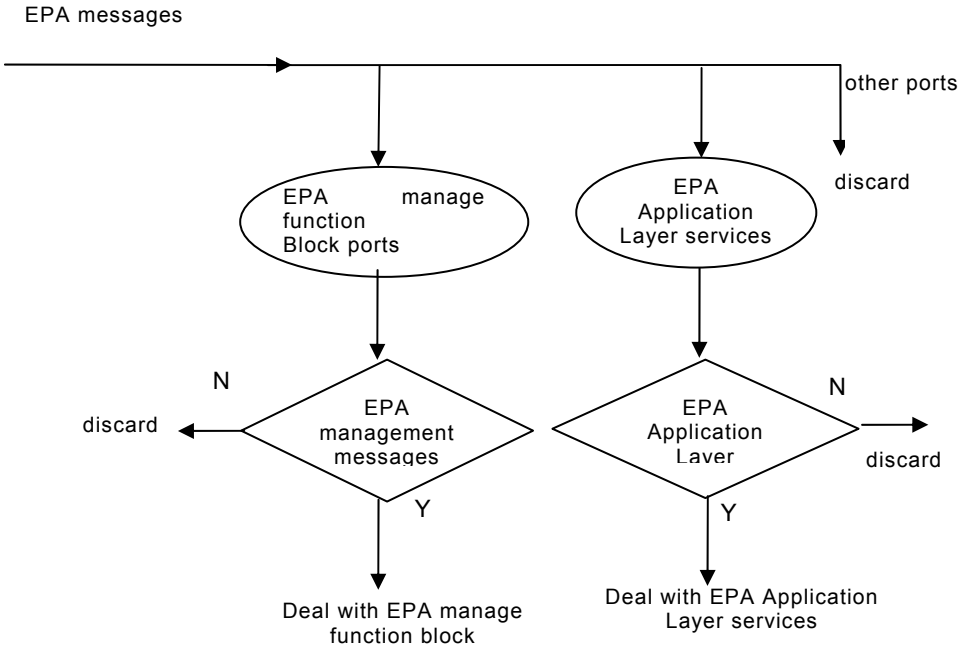


Figure 18 – received EPA messages processing procedure

8.3.7.2 EPA Socket Mapping Model Class Specification

8.3.7.2.1 EPA Socket Mapping Manager Class

8.3.7.2.1.1 Formal model

EPA ASE: **EPA SOCKET MAPPING ASE**
CLASS: **EPA SOCKET MAPPING**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1. (m) attribute: Local IP Address
2. (m) attribute: Remote IP Address
3. (m) attribute: Active Udp Port
4. (m) attribute: Active Service ID
5. (m) attribute: Active Message Length
6. (m) attribute: Active Message ID
7. (m) attribute: Active Message Time
8. (m) attribute: Active Data Pointer
9. (m) attribute: Max Message Length
10. (m) attribute: Max Retransmit Number

8.3.7.2.1.2 Attributes

Local IP Address

This attribute indicates the IP address of the local EPA device.

Remote IP Address

This attribute indicates the IP address of the remote EPA device.

Active Udp Port

This attribute indicates the UDP port to send messages.

Active Service ID

This attribute indicates which service is used to send the message.

Active Message Length

This attribute indicates the length of active message.

Active Message ID

This attribute indicates the identifier of active message.

Active Message Time

This attribute indicates the max response time for active message. It should be set 0 if it is needless.

Active Data Pointer

This attribute indicates the pointer to active message header.

Max Message Length

This attribute indicates the max permitted length of message. It will refuse to send the message whose length is beyond the value of this attribute and return an error report.

Max Retransmit Number

This attribute indicates the max times of retransmission.

8.3.7.2.1.3 Services

There are no operational services defined for this type of object.

8.3.7.2.2 EPA Socket Mapping Timer Class

8.3.7.2.2.1 Formal model

EPA ASE: **EPASocketMapping ASE**
CLASS: **EPASocketTimer**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1.	(m)	key attribute:	TimerID
2.	(m)	attribute:	Active Service ID
3.	(m)	attribute:	Active Message ID

8.3.7.2.2.2 Attributes

TimerID

This attribute indicates the identifier of the Timer.

Active Service ID

This attribute indicates which service is used to send the message.

Active Message ID

This attribute indicates the identifier of current message.

8.3.7.2.2.3 Services

There are no operational services defined for the type object.

8.4 Summary of Services in EPA application layer

Summary of services supported in application layer is listed in Table 48.

Table 48 – Summary of Services in application layer

Index	Service Name	ServiceID	Confirmed/ Unconfirmed	Priority	Description of Service
1	EM_FindTagQuery	1	Unconfirmed	2	EPA Device Query
2	EM_FindTagReply	2	Confirmed	2	EPA Device Query Reply
3	EM_GetDeviceAttribute	3	Confirmed	2	Get EPA Device Attribute
4	EM_DeviceAnnunciation	4	Unconfirmed	2	EPA Device Annunciation
5	EM_SetDeviceAttribute	5	Confirmed	2	Set EPA Device Attribute
6	EM_ClearDeviceAttribute	6	Confirmed	2	Clear EPA Device Attribute
7	DomainDownload	10	Confirmed	2	Domain Download
8	DomainUpload	11	Confirmed	2	Domain Upload
9	Read	12	Confirmed	2	Variable read
10	Write	13	Confirmed	2	Variable write
11	Distribute	14	Unconfirmed	0	Variable Distribute
12	EventNotification	15	Confirmed	1	Event notification
13	AcknowledgeEventNotification	16	Confirmed	1	Confirmed Acknowledge Event Notification
14	AlterEventConditionMonitor	17	Confirmed	1	Alter Event Condition Monitor
15	SNTP Service		Unconfirmed	3	
16	ARP Service			1	
17	RARP, ICMP, IGMP, DHCP Service			4	
18	FTP, TFTP, HTTP Services			5	

9 EPA application layer protocol specification

9.1 Syntax description

9.1.1 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
}
```

9.1.2 Confirmed request service

```
Confirmed- Request : : = CHOICE {
    EM_GetDeviceAttribute          [0]   IMPLICIT  EM_GetDeviceAttribute-RequestPDU,
    EM_SetDeviceAttribute          [1]   IMPLICIT  EM_SetDeviceAttribute-RequestPDU,
    EM_ClearDeviceAttribute        [2]   IMPLICIT  EM_ClearDeviceAttribute-RequestPDU,
    DomainDownload                 [3]   IMPLICIT  DomainDownload-RequestPDU,
    DomainUpload                   [4]   IMPLICIT  DomainUpload-RequestPDU,
    AcknowledgeEventNotification   [5]   IMPLICIT  AcknowledgeEventNotifi-RequestPDU,
    AlterEventConditionMonitor     [6]   IMPLICIT  AlterEventConditionMon-RequestPDU,
    Read                           [7]   IMPLICIT  Read-RequestPDU,
    Write                          [8]   IMPLICIT  Write-RequesPDU
}
```

9.1.3 Confirmed response service

```
Confirmed- Response : : = CHOICE {
    EM_GetDeviceAttribute          [0]  IMPLICIT  EM_GetDeviceAttribute-ResponsePDU,
    EM_SetDeviceAttribute          [1]  IMPLICIT  EM_SetDeviceAttribute-ResponsePDU,
    EM_ClearDeviceAttribute        [2]  IMPLICIT  EM_ClearDeviceAttribute-ResponsePDU,
    DomainDownload                 [3]  IMPLICIT  DomainDownload-ResponsePDU,
    DomainUpload                   [4]  IMPLICIT  DomainUpload-ResponsePDU,
    AcknowledgeEventNotification   [5]  IMPLICIT  AcknowledgeEventNotifi-ResponsePDU,
    AlterEventConditionMonitor     [6]  IMPLICIT  AlterEventConditionMon-ResponsePDU,
    Read                           [7]  IMPLICIT  Read-ResponsePDU,
    Write                          [8]  IMPLICIT  Write-ResponsePDU
}
```

9.1.4 Confirmed error

```
Confirmed- Error : : = CHOICE {
    EM_GetDeviceAttribute          [0]   IMPLICIT  Error-Type,
    EM_SetDeviceAttribute          [1]   IMPLICIT  Error-Type,
    EM_ClearDeviceAttribute        [2]   IMPLICIT  Error-Type,
    DomainDownload                 [3]   IMPLICIT  Error-Type,
    DomainUpload                   [4]   IMPLICIT  Error-Type,
    AcknowledgeEventNotification   [5]   IMPLICIT  Error-Type,
    AlterEventConditionMonitor     [6]   IMPLICIT  Error-Type,
    Read                           [7]   IMPLICIT  Error-Type,
    Write                          [8]   IMPLICIT  Error-Type
}
```

9.1.5 Error type

```
ErrorType : : = SEQUENCE {
    ErrorClass             [0]          IMPLICIT  Integer8,
    ErrorCode              [1]          IMPLICIT  Integer8,
    AdditionalCode         [2]          IMPLICIT  Integer8,
    Reserved               [3]          IMPLICIT  OctetString,
    AdditionalDescription  [4]          IMPLICIT  VisibleString
}
```

9.1.6 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)
}			
}			

9.1.7 Unconfirmed request

```
Unconfirmed-Request: : = CHOICE {

    EM_FindTagQuery          [0] IMPLICIT EM_FindTagQuery-RequestPDU,
    EM_FindTagReply          [1] IMPLICIT EM_FindTagReply-RequestPDU,
    EM_DeviceAnnunciation    [2] IMPLICIT EM_DeviceAnnunciation-RequestPDU,
    EventNotification        [3] IMPLICIT EventNotification-RequestPDU,
    Distribute               [4] IMPLICIT Distribute-RequestPDU

}
```

9.1.8 EPA application layer PDU

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

9.1.9 APDU header format

```
PDUHeader : : = SEQUENCE {

    ServiceID          [0]          IMPLICIT Unsigned8,
    Reserved           [1]          IMPLICIT OctetString,
    Length             [2]          IMPLICIT Unsigned16,
    MessageID          [3]          IMPLICIT Unsigned16

}
```

9.1.10 EPA System Manage Entity services

9.1.10.1 EM_FindTagQuery service

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

9.1.10.2 EM_FindTagReply service

```
EM_FindTagReply -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
}
```

9.1.10.3 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
}
```

9.1.10.4 EM_DeviceAnnunciation service

```
EM_DeviceAnnunciation-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
}
```

9.1.10.5 EM_SetDeviceAttribute service

```
EM_SetDeviceAttribute-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_SetDeviceAttribute-ResponsePDU : : = CHOICE {
    EM_SetDeviceAttribute-PositiveResponsePDU,
    EM_SetDeviceAttribute-NegativeResponsePDU
}

EM_SetDeviceAttribute-PositiveResponsePDU : : = SEQUENCE {
    DestinationIPAddress        [0]   IMPLICIT  Unsigned32,
    MaxRedundancyNumber         [1]   IMPLICIT  Unsigned8
}

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

9.1.10.6 EM_ClearDeviceAttribute service

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

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

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

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

9.1.11 EPA Application Access Entity (AAE) services

9.1.11.1 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
}
```

9.1.11.2 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
}
```

9.1.11.3 EventNotification service

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

9.1.11.4 AcknowledgeEventNotification service

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

AcknowledgeEventNotification -ResponsePDU : : = CHOICE {
    AcknowledgeEventNotification -PositiveResponsePDU,
    AcknowledgeEventNotification -NegativeResponsePDU
}
```

```
AcknowledgeEventNotification -PositiveResponsePDU: : = SEQUENCE{

    DestinationAppID        [0]  IMPLICIT  Unsigned16

}

AcknowledgeEventNotification -NegativeResponsePDU: : = SEQUENCE{

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

}
```

9.1.11.5 AlterEventConditionMonitor service

```
AlterEventConditionMonitor-RequestPDU : : = SEQUENCE {

    DestinationAppID        [0]  IMPLICIT  Unsigned16,
    DestinationObjectID     [1]  IMPLICIT  Unsigned16,
    Enabled                 [2]  IMPLICIT  Boolean

}

AlterEventConditionMonitor -ResponsePDU : : = CHOICE {

    AlterEventConditionMonitor -PositiveResponsePDU,
    AlterEventConditionMonitor -NegativeResponsePDU

}

AlterEventConditionMonitor -PositiveResponsePDU: : = SEQUENCE{

    DestinationAppID        [0]  IMPLICIT  Unsigned16

}

AlterEventConditionMonitor -NegativeResponsePDU: : = SEQUENCE{

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

}
```

9.1.11.6 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
}
```

9.1.11.7 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
}
```

9.1.11.8 Distribute service

```
Distribute-RequestPDU : : = SEQUENCE {
    SourceAppID             [0]  IMPLICIT  Unsigned16,
    SourceObjectID          [1]  IMPLICIT  Unsigned16,
    Data                    [2]  IMPLICIT  OctetString
}
```

9.1.12 Abstract syntax of data type

9.1.12.1 Notation of Boolean type

```
Boolean ::= BOOLEAN                 --value is non-zero means TRUE.

                                    --value is zero means FALSE.
```

9.1.12.2 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 $-2^{63} <= i <= 2^{63}-1$

9.1.12.3 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 $0 <= i <= 2^{64}-1$

9.1.12.4 Notation of float data type

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

9.1.12.5 Notation of visible string type

VisibleString ::= VISIBLE STRING --general use

9.1.12.6 Notation of octet string type

OctetString ::= OCTET STRING --general use

9.1.12.7 Notation of bit string type

BitString ::= BIT STRING -- general use

9.1.12.8 Notation of TimeofDay type

TimeOfDay ::= OctetString6

9.1.12.9 Notation of binary date type

BinaryDate ::= OctetString8

9.1.12.10 Notation of TimeDifference type

TimeDifference ::= OctetString6

9.2 Transfer Syntax

9.2.1 Encodingof basic data types

9.2.1.1 Boolean

A Boolean value is encoded in one byte, 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 below (Bit 7 is MSB, Bit 0 is LSB).

Table 49 – Encoding of Boolean value TRUE

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	1	1	1	1	1	1	1	1

If the value is FALSE, the value of each bit is shown in the table below.

Table 50 – Encoding of Boolean value FALSE

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	0	0	0	0	0	0	0	0

9.2.1.2 Unsigned8

The Unsigned8 type is encoded in one byte, the range is 0~255, the weight of each bit is shown in the table below.

Table 51 – Encoding of Unsigned8 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

9.2.1.3 Unsigned16

The Unsigned16 type is encoded in two bytes, the range is 0~216-1, the weight of each bit is shown in the table below.

Table 52 – Encoding of Unsigned16 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
Byte2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

9.2.1.4 Unsigned32

The Unsigned32 type is encoded in four bytes, the range is 0~232-1, the weight of each bit is shown in the table below.

Table 53 – Encoding of Unsigned32 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
Byte4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

9.2.1.5 Unsigned64

The Unsigned64 type is encoded in eight bytes, the range is 0~264-1, the weight of each bit is shown in the table below.

Table 54 – Encoding of Unsigned64 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Byte2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Byte3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
Byte4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Byte5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.6 Int8

The Int8 type is encoded in one byte, 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 the table below.

Table 55 – Encoding of Int8 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	SN	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.7 Int16

The Int16 type is encoded in two bytes, the range is -2^{15}~2^{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 the table below.

Table 56 – Encoding of Int16 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.8 Int32

The Int32 type is encoded in four bytes, the range is -2^{31}~2^{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 the table below.

Table 57 – Encoding of Int32 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.9 Int64

The Int64 type is encoded in eight bytes, the range is -2^{63}~2^{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 the table below.

Table 58 – Encoding of Int64 data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Byte2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Byte3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
Byte4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Byte5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.10 Real

The Real type is encoded in four bytes. 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 byte 1 and bit 7 of byte 2 defines the exponent field. It is followed by the fraction field from bit 6 of byte 1 to bit 0 of byte 2. The weight of each bit is shown in the table below.

Table 59 – Encoding of Real type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
Byte2	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Byte3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
Byte4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

9.2.1.11 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 the table below.

Table 60 – Encoding of VisibleString data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	the first character							
Byte2	the second character							
……	and so on……							
……	so on……							
Byten								

9.2.1.12 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 the table below.

Table 61 – Encoding of OctetString data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	Binary data							
Byte2	Binary data							
……	……							
……	……							
Byten								

9.2.1.13 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 the table below.

Table 62 – Encoding of BitString data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	0	1	2	3	4	5	6	7
Byte2	8	9	10	11	12	13	14	15
……	so on……							
……	so on……							
Byten								

9.2.1.14 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 (byte 1 and byte 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 byte 3 to byte 6). The encoding is shown in the table below.

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

Table 63 – Encoding of TimeOfDay data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Byte3	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte4	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.15 BinaryDate

The type BinaryDate consists of a calendar date and a time, it is encoded in 8 octets shown in the table below.

The year field is encoded as an Unsigned16 data (byte1 and byte2), if its value is 2004, it represents the year of 2004.

The month field is encoded as an Unsigned8 data (byte 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 (byte 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 (byte5), the range is 0~23, its value represents one hour of 24 hours in a day.

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

The millisecond field is encode as an Unsigned16 data (byte7 and byte 8), the range is 0~59999, its value represents one millisecond of 60000 milliseconds in a minute.

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

Table 64 – Encoding of BinaryDate data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Byte3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Byte4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Byte5	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Byte6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Byte7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.16 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 (byte 1 and byte 2), its value represents the days difference.

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

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

Table 65 – Encoding of TimeDifference data type

	B7	B6	B5	B4	B3	B2	B1	B0
Byte1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Byte3	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte4	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

9.2.1.17 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.

9.2.1.18 Encoding of CHOICE

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

9.2.2 Object definitions in EPA System Management ASE

9.2.2.1 EPA MIB Header

The object of EPA MIB Header is defined as shown in the table below.

Table 66 – Definition of EPA MIB 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 MIB header object in the EPA MIB
2	MIB Revision Number	Read Only	Unsigned16	2	2	The version of EPA MIB

9.2.2.2 EPA Device Descriptor Object

The object of EPA Device Descriptor Object is defined is show in the table below.

Table 67 – 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 MIB
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

9.2.2.3 Time Synchronization Object

The object of Time Synchronization Object class is defined as shown in the table below:

Table 68 – Definition of 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 Time Synchronization Object in the MIB
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

9.2.2.4 Maximum Response Time Object

The object of Maximum Response Time Object class is defined as shown in the table below.

Table 69 – 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 MIB
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

9.2.2.5 EPA Communication Scheduling Management Object

The object of EPA Communication Scheduling Management class is defined as shown in the table below.

Table 70 – Definition of 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 MIB
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

9.2.2.6 Device Application Information Object

The object of Device Application Information class is defined as shown in the table below.

Table 71 – Definition of 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 MIB
2	XDDL Version	Read Only	Unsigned16	2	2	The device description version number

9.2.2.7 FB Application Information Header

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

Table 72 – 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 MIB
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

9.2.2.8 Domain Application Information Header

The object of Domain Application Information Header class is defined as shown in the table below.

Table 73 – 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 MIB
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 MIB
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

9.2.2.9 EPA Link Object Header

The object of EPA Link Object Header class is defined as shown in the table below.

Table 74 – 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 MIB
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 MIB
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

9.2.2.10 FB Application Information Object

The object of FB Application Information class is defined as shown in the table below.

Table 75 – 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 MIB
2	Reserved	Read Only	Unsigned16	2	2	Reserved
3	FB Name	Read Only	VisibleString	4	32	FB Name, its length is 32 bytes, if the string length is less than 32 bytes, 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

9.2.2.11 EPA Link Object

The object of EPA Link Object class is defined as shown in the table below.

Table 76 – 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 MIB
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 bytes of TimeDifference.

9.2.2.12 Domain Application Information Object

The object of Domain Application Information class is defined as shown in the table below.

Table 77 – 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 MIB
2	Domain Object ID	Read Only	Unsigned16	2	2	The index of 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 bytes, the unused part is padded with BLANK (0x20).

9.2.3 Definition of objects used in EPA application access entity

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

9.2.3.1 Domain Object

The object of Domain class is defined as shown in the table below.

Table 78 – 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 bytes 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

No.	Parameter name	Read/write Property	Data type	Octet offset	Octet length
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 can't be overwritten by the download service, its data type is unsigned16

9.2.3.2 Simple Variable Object

Definition of Simple Variable Object is shown in the table below.

Table 79 – 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 MIB
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 byte
4	Local Address	Read Only	Unsigned32	4	The pointer pointed to the specific Variable Object which can be use 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

9.2.3.3 Event Object

Definition of Event Object is shown in the table below.

Table 80 – 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 byte 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.

9.2.3.4 EPA Socket Mapping Object

Definition of EPA Socket Mapping Object is shown in the table below.

Table 81 – 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

No.	Parameter name	Read/write Property	Data type	Octet offset	Octet length
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 don't 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 exceeded 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.

9.2.3.5 EPA Socket Timer Object

Definition of EPA Socket Timer Object is shown in the table below.

Table 82 – 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 don't need response, it should be set to zero. The unit is millisecond.

9.2.3.6 ErrorType Object

Definition of ErrorType Object is shown in the table below.

Table 83 – 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

9.2.4 Encoding of EPA APDU Header

The encoding of EPA Application layer Service Message packet header is shown in the table below.

Table 84 – 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, see Table 48.
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.

9.2.5 Encoding of EPA System Management Entity service parameters

9.2.5.1 EM_FindTagQuery service

EM_FindTagQuery request parameters are coded as shown in Table 85.

Table 85 – Encoding of EM_FindTagQuery 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 bytes, and the unused part is padded with BLANK (0x20).
4	FB Tag	VisibleString	36	32	Functions 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.

9.2.5.2 EM_FindTagReply service

EM_FindTagReply request parameters are coded as shown in the table below.

Table 86 – Encoding of EM_FindTagReply 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

No.	Parameter name	Data type	Octet offset	Octet length	Description
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 bytes, 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 bytes, and the unused part is padded with BLANK (0x20).

9.2.5.3 EM_GetDeviceAttribute service

9.2.5.3.1 Request primitive

EM_GetDeviceAttribute request parameters are coded as shown in Table 87.

Table 87 – 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

9.2.5.3.2 Positive response primitive

EM_GetDeviceAttribute positive response parameters are coded as shown in Table 88.

Table 88 – 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 bytes, 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

No.	Parameter name	Data type	Octet offset	Octet length	Description
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——backup status If the redundancy number is 0, then the response primitive doesn't 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 doesn't 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 doesn't contain this parameter.

9.2.5.3.3 Negative response primitive

EM_GetDeviceAttribute negative response parameters are coded as shown in Table 89.

Table 89 – 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

9.2.5.4 EM_DeviceAnnunciation service

EM_DeviceAnnunciation request parameters are coded as shown in Table 90.

Table 90 – Encoding of EM_DeviceAnnunciation 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 bytes, 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 active device, then this value is 0, otherwiase the following parameters 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

No.	Parameter name	Data type	Octet offset	Octet length	Description
					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 request the service
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 doesn't contain this parameter.

9.2.5.5 EM_SetDeviceAttribute service

9.2.5.5.1 Request Primitive

EM_SetDeviceAttribute request parameters are coded as shown in Table 91.

Table 91 – Encoding of EM_SetDeviceAttribute 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 bytes, 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

No.	Parameter name	Data type	Octet offset	Octet length	Description
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 local device, if its active device, then this value is 0, and doesn't have the following parameters.
7	LAN Redundancy Port	Unsigned16	72	2	The LAN redundancy message processing port of the device which request the service
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.

9.2.5.5.2 Positive response primitive

EM_SetDeviceAttribute positive response parameters are coded as shown in Table 92.

Table 92 – Encoding of EM_SetDeviceAttribute 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

9.2.5.5.3 Negative response primitive

EM_SetDeviceAttribute negative response parameters are coded as shown in Table 93.

Table 93 – Encoding of EM_SetDeviceAttribute 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

9.2.5.6 EM_ClearDeviceAttribute service

9.2.5.6.1 Request primitive

EM_ClearDeviceAttribute request parameters are coded as shown in Table 94.

Table 94 – Encoding of EM_ClearDeviceAttribute 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 bytes, and the unused part is padded with BLANK (0x20).

9.2.5.6.2 Positive response primitive

EM_ClearDeviceAttribute positive response parameters are coded as shown in Table 95.

Table 95 – Encoding of EM_ClearDeviceAttribute positive response parameters

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

9.2.5.6.3 Negative response primitive

EM_ClearDeviceAttribute negative response parameters are coded as shown in Table 96.

Table 96 – Encoding of clear Clear Device Attribute Service Refuse Packet

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

9.2.6 Encoding of AAE Services

9.2.6.1 DomainDownload Service

9.2.6.1.1 Request primitive

DomainDownload request parameters are coded as shown in Table 97.

Table 97 – 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

9.2.6.1.2 Positive response primitive

DomainDownload positive response parameters are coded as shown in Table 98.

Table 98 – 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

9.2.6.1.3 Negative response primitive

DomainDownload negative response parameters are coded as shown in Table 99.

Table 99 – 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

9.2.6.2 Domain Upload Service

9.2.6.2.1 Request primitive

DomainUpload request parameters are coded as shown in Table 100.

Table 100 – 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

9.2.6.2.2 Positive response primitive

DomainUpload positive response parameters are coded as shown in Table 101.

Table 101 – 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

9.2.6.2.3 Negative response primitive

DomainUpload negative response parameters are coded as shown in Table 102.

Table 102 – 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

9.2.6.3 EventNotification Service

EventNotification request parameters are coded as shown in Table 103.

Table 103 – Encoding of EventNotification 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

9.2.6.4 EventNotificationAcknowledge Service

9.2.6.4.1 Request primitive

EventNotificationAcknowledge request parameters are coded as shown in Table 104.

Table 104 – Encoding of EventNotificationAcknowledge 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

9.2.6.4.2 Positive response primitive

EventNotificationAcknowledge positive response parameters are coded as shown in Table 105.

Table 105 – Encoding of EventNotificationAcknowledge positive response parameters

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

9.2.6.4.3 Negative response primitive

EventNotificationAcknowledge negative response parameters are coded as shown in Table 106.

Table 106 – Encoding of EventNotificationAcknowledge 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

9.2.6.5 AlterEventConditionMonitor service

9.2.6.5.1 Request primitive

AlterEventConditionMonitor request parameters are coded as shown in Table 107.

Table 107 – Encoding of AlterEventConditionMonitor 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

9.2.6.5.2 Positive response primitive

AlterEventConditionMonitor positive response parameters are coded as shown in Table 108.

Table 108 – Encoding of AlterEventConditionMonitor positive response parameters

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

9.2.6.5.3 Negative response primitive

AlterEventConditionMonitor negative response parameters are coded as shown in Table 109.

Table 109 – Encoding of AlterEventConditionMonitor 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

9.2.6.6 Read service

9.2.6.6.1 Request primitive

Read request parameters are coded as shown in Table 110.

Table 110 – 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

9.2.6.6.2 Positive response primitive

Read positive response parameters are coded as shown in Table 111.

Table 111 – 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

9.2.6.6.3 Negative response primitive

Read negative response parameters are coded as shown in Table 112.

Table 112 – 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

9.2.6.7 Write Service

9.2.6.7.1 Request primitive

Write request parameters are coded as shown in Table 113.

Table 113 – 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

9.2.6.7.2 Positive response primitive

Write positive response parameters are coded as shown in Table 114.

Table 114 – 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

9.2.6.7.3 Negative response primitive

Write negative response parameters are coded as shown in Table 115.

Table 115 – 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

9.2.6.8 Distribute Service

Distribute request parameters are coded as shown in Table 116.

Table 116 – Encoding of Distribute 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	Distributed data

9.3 Protocol State Machine

Figure 19 illustrates the relationships of primitives. The primitives exchanged between SME and application layer user is illustrated in 9.4.1.1. The primitives exchanged between SME and ESME is illustrated in 9.4.1.3. The primitives exchanged between AAE and application layer user is illustrated in 9.5.1.1. The primitives exchanged between AAE and ESME is illustrated in 9.5.1.3. The primitives exchanged between Transport Layer and ESME is illustrated in 9.6.1.5.

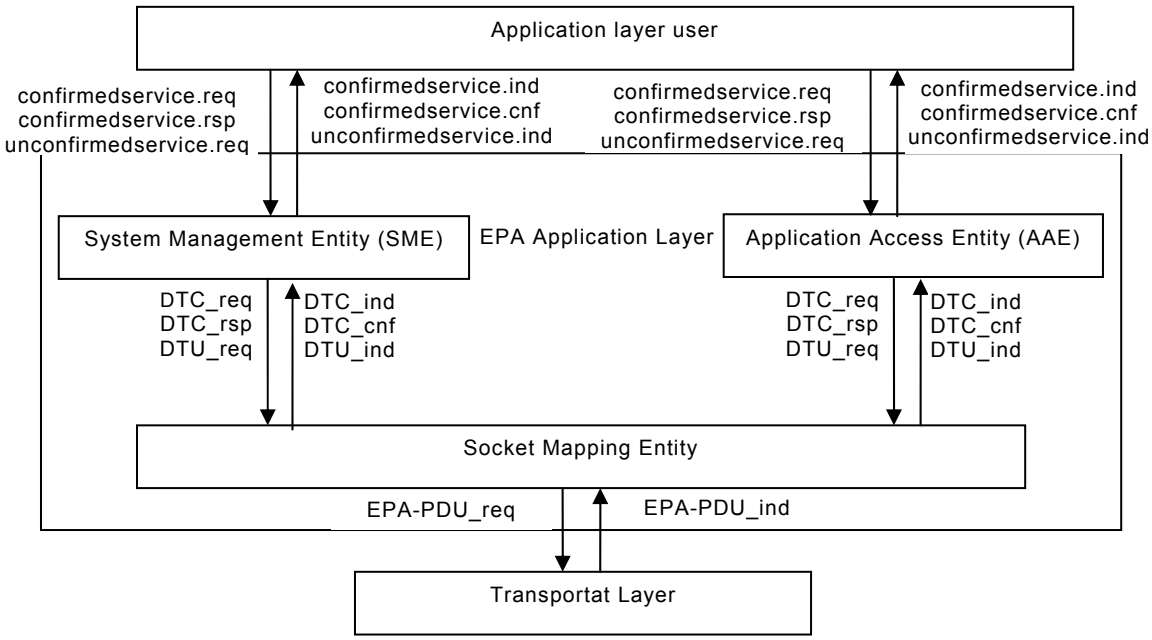


Figure 19 – Exchanged Primitives of Protocol State Machine

9.4 EPA SME State Machines

9.4.1 Primitives

9.4.1.1 Primitives exchanged between SME and application layer user

Table 117 and Table 118 show the primitives exchanged between SME and application layer **user.**

Table 117 – Primitives delivered by application layer user to SME

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 SME 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 SME 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 SME by the user of application layer.

Table 118 – Primitives delivered by SME to application layer user

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

9.4.1.2 Primitive parameters of SME and user of application layer

Table 119 shows the primitives parameters of SME and the user of application layer.

Table 119 – Primitive parameters exchanged between SME and application layer user

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

9.4.1.3 Primitives exchanged between SME and ESME

Table 120 and table 121 show the primitives exchanged between SME and ESME.

Table 120 – Primitives delivered by SME to ESME

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

Table 121 – Primitives delivered by ESME to SME

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

9.4.1.4 Primitive parameters exchanged between SME and ESME

Table 122 shows the primitives parameters exchanged between SME and ESME.

Table 122 – Primitives parameters exchanged between SME and ESME

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

9.4.2 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 20.

9.4.2.1 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.

9.4.2.2 Unconfigured

In this state, EPA System Management Entity (SME) uses the specific multicast destination IP address to send EM_DeviceAnnunciation request message to inform other devices its presence. When received this request message, user configuration application can query, clear or set the attributes of this device using EM_FindTagQuery, EM_SetDeviceAttribute and EM_ClearDeviceAttribute services. After configuration, EPA SME will change its state into *Configured* and start normal operation.

9.4.2.3 Configured

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

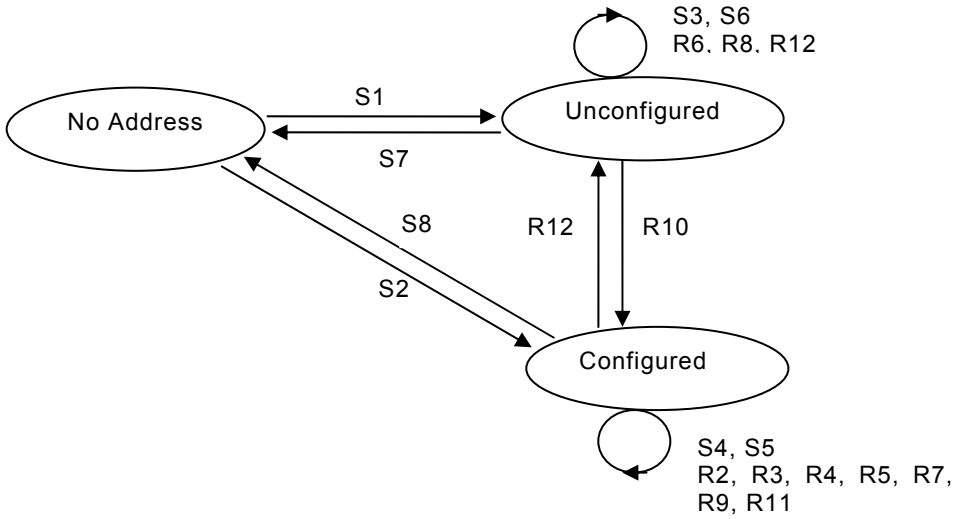


Figure 20 – Protocol State Machine of EPA SME

9.4.3 State Transitions

The state transitions of EPA SME are defined in Table 123.

Table 123 – State Transitions of EPA SME

#	Current State	Event & Action => Action	Next State
S1	No Address	EpaRcvNewIpAddress (address) = TRUE && EpaAttribute_Set ()= FALSE => EpaRestoreDefaults () EpaNewAddress(address) EM_DeviceAnnunciation.req {} Restart_EPARepeatTimer()	Unconfigured
S2	No Address	EpaRcvNewIpAddress (address) = TRUE && EpaAttribute_Set()=TRUE => EpaClear_DuplicatePdTagFlag() EpaNewAddress (address) EM_DeviceAnnunciation.req {} Restart_EPARepeatTimer () EM_FindTagQuery.req {}	Configured
S3	Unconfigured	EpaRepeatTimerExpires () => EM_DeviceAnnunciation.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

#	Current State	Event & Action => Action	Next State
S6	Unconfigured	EpaSntpSyncLost () => EM_DeviceAnnunciation.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_SME Rsp Message” || EpaRecvMsg ()=”Any Confirmed EPA_SME Error Message” => EM_ConfirmedService.cnf{}	Unconfigured
R2	Configured	EpaRecvMsg()=” EM_FindTagQuery” && EpaQueryMatch (em_svc) = TRUE => EM_FindTagReply.req {}	Configured
R3	Configured	EpaRecvMsg()=” EM_FindTagReply” && Epa_MessageIdMatch(em_svc) = TRUE && EpaDeviceId_Match (em_svc) = FALSE => EPASet_DuplicatePdTagFlag () EM_DeviceAnnunciation.req {} Restart_EPARepeatTimer ()	Configured
R4	Configured	EpaRecvMsg()=” EM_FindTagReply” && Epa_MessageIDMatch(em_svc) = TRUE && EpaDeviceId_Match (em_svc) = TRUE => //Do nothing-the response is from this device	Configured
R5	Configured	EpaRecvMsg() = ” EM_FindTagReply” => EM_FindTagReply.ind {}	Configured
R6	Unconfigured	EpaRecvMsg() = ” EM_GetDeviceAttributeReq” => EM_ConfrimedService.err {}	Unconfigured
R7	Configured	EpaRecvMsg()=” EM_GetDeviceAttributeReq” => EM_GetDeviceAttribute.rsp {}	Configured
R8	Unconfigured	EpaRecvMsg()=” EM_SetDeviceAttributeReq” || EpaRecvMsg()=” EM_ClearDeviceAttributeReq” && EpaDeviceId_Match(em_svc)= FALSE => EM_ConfirmedService.err { }	Unconfigured

#	Current State	Event & Action => Action	Next State
R9	Configured	EpaRecvMsg()=”EM_SetDeviceAttributeReq” || EpaRecvMsg()=” EM_ClearDeviceAttributeReq” && EpaPdTag_Match(em_svc)= TRUE => EM_ConfirmedService.rsp {}	Configured
R10	Unconfigured	EpaRecvMsg()=” EM_SetDeviceAttributeReq” => EpaSet_Attribute_Data(em_svc) EpaClear_DuplicatePdTagFlag () EM_SetDeviceAttribute.rsp {} EM_DeviceAnnunciation.req {} Restart_EPARepeatTimer () EM_FindTagQuery.req {}	Configured
R11	Configured	EpaRecvMsg=” EM_GetDeviceAttributeReq” => EM_GetDeviceAttribute.rsp {}	Configured
R12	Configured	EpaRecvMsg()=”EM_ClearDeviceAttributeReq” => EpaResoreDefaults () EM_ClearDeviceAttribute.rsp {} EM_DeviceAnnunciation.req {} Restart_EPARepeatTimer ()	Unconfigured

9.4.4 Function descriptions

The functions used in EPA SME state transitions are listed in Table 124 through Table 141.

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

9.4.4.1 EpaRcvNewIpAddress()

EpaRcvNewIpAddress() is illustrated in Table 124.

Table 124 – EpaRcvNewIpAddress() descriptions

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

9.4.4.2 EpaAttribute_Set()

EpaAttribute_Set() is illustrated in Table 125.

Table 125 – EpaAttribute_Set() descriptions

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

9.4.4.3 EpaRestoreDefaults()

EpaRestoreDefaults() is illustrated in Table 126.

Table 126 – EpaRestoreDefaults() descriptions

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

9.4.4.4 EpaNewAddress()

EpaNewAddress() is illustrated in Table 127.

Table 127 – EpaNewAddress() descriptions

Name	EpaNewAddress	Using	SME
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.			

9.4.4.5 Restart_EPARepeatTimer()

Restart_EPARepeatTimer() is illustrated in Table 128.

Table 128 – Restart_EPARepeatTimer() descriptions

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

9.4.4.6 EpaClear_DuplicatePdTagFlag()

EpaClear_DuplicatePdTagFlag() is illustrated in Table 129.

Table 129 – EpaClear_DuplicatePdTagFlag() descriptions

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

9.4.4.7 EPARepeatTimerExpire()

EPARepeatTimerExpire() is illustrated in Table 130.

Table 130 – EPARepeatTimerExpire() descriptions

Name	EPARepeatTimerExpire	Using	SME
Input		**Output**	
None		None	
Function			
The function is invoked when the EPA Repeat Timer expires			

9.4.4.8 EpaSend_EM_ReqRspMessage()

EpaSend_EM_ReqRspMessage() is illustrated in Table 131.

Table 131 – EpaSend_EM_ReqRspMessage() descriptions

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

9.4.4.9 EpaSend_EM_CommonErrorRsp()

EpaSend_EM_CommonErrorRsp() is illustrated in Table 132.

Table 132 – EpaSend_EM_CommonErrorRsp() descriptions

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

9.4.4.10 EpaSntpSyncLost()

EpaSntpSyncLost() is illustrated in Table 133.

Table 133 – EpaSntpSyncLost() descriptions

Name	EpaSntpSyncLost	Using	SME
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.			

9.4.4.11 EpaIPAddressCollision()

EpaIPAddressCollision() is illustrated in Table 134.

Table 134 – EpaIPAddressCollision() descriptions

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

9.4.4.12 EpaRecvMsg()

EpaRecvMsg() is illustrated in Table 135.

Table 135 – EpaRecvMsg() descriptions

Name	**EpaRecvMsg**	**Using**	**SME**
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.			

9.4.4.13 EpaQueryMatch()

EpaQueryMatch() is illustrated in Table 136.

Table 136 – EpaQueryMatch() descriptions

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

9.4.4.14 EpaMessageIDMatch()

EpaMessageIDMatch() is illustrated in Table 137.

Table 137 – EpaMessageIDMatch() descriptions

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

9.4.4.15 EpaDeviceId_Match()

EpaDeviceId_Match() is illustrated in Table 138.

Table 138 – EpaDevId_Match() descriptions

Name	EpaDeviceId_Match	Using	SME
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.			

9.4.4.16 EpaPdTag_Match()

EpaPdTag_Match() is illustrated in Table 139.

Table 139 – EpaPdTag_Match() descriptions

Name	EpaPdTag_Match	Using	SME
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.			

9.4.4.17 EpaSet_Attribute_Data()

EpaSet_Attribute_Data() is illustrated in Table 140.

Table 140 – EpaSet_Attribute_Data() descriptions

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

9.4.4.18 EpaSet_DuplicatePdTagFlag()

EpaSet_DuplicatePdTagFlag() is illustrated in Table 141.

Table 141 – EpaSet_DuplicatePdTagFlag() descriptions

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

9.5 Application Access Entity Protocol Machine

9.5.1 Primitives

9.5.1.1 Primitives Exchanged between AAE and application layer user

The primitives exchanged between AAE and Application Layer User (ALU) are shown in Table 142 through Table 143.

Table 142 – 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 143 – 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.

9.5.1.2 Primitive parameters exchanged between AAE and ALU

Table 144 describes the parameters used with primitives exchanged between AAE and ALU.

Table 144 – 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.

9.5.1.3 Primitives Exchanged between AAE and ESME

The primitives exchanged between AAE and ESME are shown in Table 145 through Table 146.

Table 145 – Primitives issued by AAE to ESME

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

Table 146 – Primitives issued by ESME to AAE

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

9.5.1.4 Primitive parameters exchanged between AAE and ESME

Table 147 describes the parameters used with primitives exchanged between AAE and ESME.

Table 147 – Primitive parameters exchanged between AAE and ESME

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 receive data from.
data	This parameter transfer the data that the sender will send and the receiver will receive.

9.5.2 AAE state machine

9.5.2.1 AAE state description

EPA Application Access Entity is always active. Its state is described in Table 148.

Table 148 – AAE state descriptions

States	Descriptios
ACTIVE	The ACEs in ACTIVE state prepare to transfer service primitives to ALU and ESME,or to receive primitives from ALU and ESME.

9.5.2.2 State transitions

The protocol state machine of AAE is illustrated in the Figure 21 and Table 149 through Table 150.

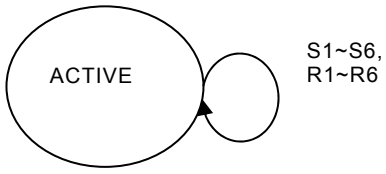


Figure 21 – AAE state transition diagram

Table 149 – AAE state transitions (sender)

#	Current state	Event or condition => action	Next state

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

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

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

S4	ACTIVE	ConfirmedService.req
		=>
		DTC.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S5	ACTIVE	ConfirmedService.rsq
		=>
		DTC.rsq {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

S6	ACTIVE	UnconfirmedService.req
		=>
		DTU.req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

Table 150 – AAE state transitions (receiver)

#	Current state	Event or condition => action	Next state

R1	ACTIVE	Confirmedservice.ind
		=>
		ConfirmedService.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
		
			ACTIVE

R2	ACTIVE	Confirmedservice.cnf
		=>
		ConfirmedService.cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
		
			ACTIVE

R3	ACTIVE	UnconfirmedService.ind
		=>
		UnconfirmedService.ind {
		Data := user_data,
		Destination_ip : = remote_ip_address,
		}
		
			ACTIVE

R4	ACTIVE	DTC.ind
		&& ServiceType(data) = “Confirmed Service Indication”
		=>
		ConfirmedService.ind {
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

R5	ACTIVE	DTC.cnf
		&& ServiceType(data) = “Confirmed Service Confirmation”
		=>
		ConfirmedService.cnf {
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

R6	ACTIVE	DTU.ind
		&& ServiceType(data) = “Unconfirmed Service Indication”
		=>
		UnconfirmedService.ind {
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

9.5.2.3 Functions descriptions

Table 151 describes the functions used by AAE state transitions.

Table 151 – ServiceType() descriptions

Name	ServiceType	Used in	AAE
Input		**Output**	
data		Receive the primitive type of service message	
Function			
To Judge the message type by receiving service message,including confirm service indication primitive,confirm service primitive and nonconfirm service indication primitive.			

9.5.3 Event ASE Protocol Machine

9.5.3.1 State description

Event ASE has two states: LOCKED and UNLOCKED described in Table 152.

Table 152 – State value of event management

States	Descriptios
LOCKED	LOCKED(Enabled=FALSE)means no event notification transfer
UNLOCKED	UNLOCKED(Enable = TRUE)means event notification transfer normally

9.5.3.2 State transitions

State transitions of Event ASE are shown in Figure 22 and Table 153 .

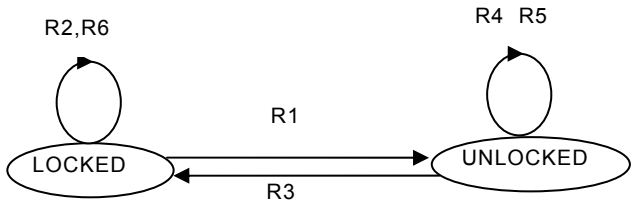


Figure 22 – Event ASE state transition diagram

Table 153 – Event ASE state transition table

#	Current state	Event or condition => action	Next state

R1	LOCKED	AlterEventConditionMonitor.ind
		&& Enabled = TRUE
		=>
		AlterEventConditionMonitor.rsq(+){
		}
			UNLOCKED

R2	LOCKED	AlterEventConditionMonitor.ind
		&&Enabled = FALSE
		=>
		AlterEventConditionMonitor.rsq(+){
		}
			LOCKED

R3	UNLOCKED	AlterEventConditionMonitor.ind
		&&Enabled = FALSE
		=>
		AlterEventConditionMonitor.rsq(+){
		}
			LOCKED

R4	UNLOCKED	AlterEventConditionMonitor.ind
		&&Enabled = TRUE
		=>
		AlterEventConditionMonitor.rsq(+){
		}
			UNLOCKED

R5	UNLOCKED	EPA AL service.ind <>” AlterEventConditionMonitor.ind” => (no actions taken)	UNLOCKED
R6	LOCKED	EPA AL service.ind <>” AlterEventConditionMonitor.ind” => (no actions taken)	LOCKED

9.5.3.3 Function descriptions

No additional functions are used in Event ASE state transitions.

9.5.4 Domain ASE Protocol Machine

9.5.4.1 State descriptions

Domain ASE has five states described in Table 154.

Table 154 – Domain state value

Domain state code	value	specification
EXISTENT	0	This domain exists but its content is not defined
DOWNLOADING	1	This domain is being downloaded
UPLOADING	2	This domain is being uploaded
READY	3	This domain can be used
IN-USE	4	This domain is in use by some program. It cannot be downloaded or uploaded in this state

9.5.4.2 State transitions

The state transitions of Domain ASE are illustrated in Figure 23 and Table 155.

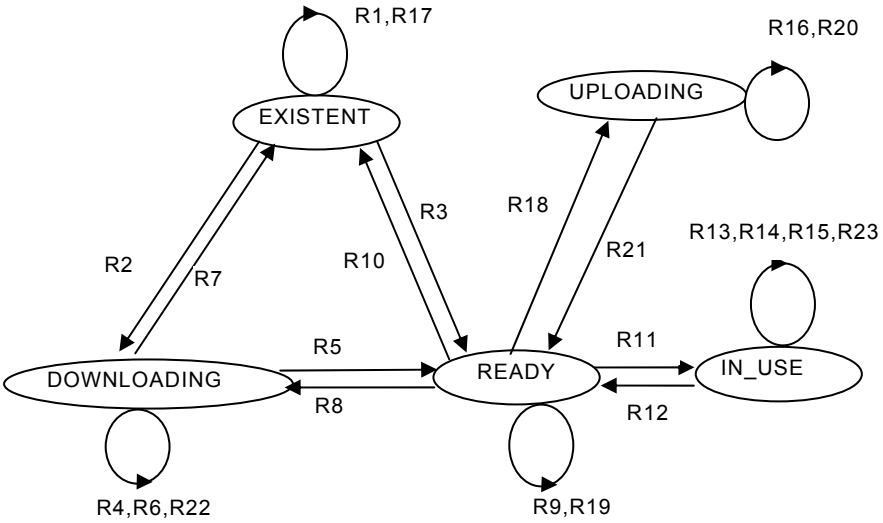


Figure 23 – Domain ASE state transition diagram

Table 155 – Domain ASE state transition table

	Current state	Event or condition => action	Next state

R1	EXISTENT	DomainDownload.ind
		&& Domain_DownloadSucceed() = FALSE
		=>
		DomainDownload.err{
		}
			EXISTENT

R2	EXISTENT	DomainDownload.ind
		&& Domain_DownloadSucceed() = TRUE
		&&MoreFollows = TRUE
		=>
		DomainDownload.rsp(+){
		}
		Domain_WriteBuffer()
			DOWNLOADING

R3	EXISTENT	DomainDownload.ind
		&& Domain_DownloadSucceed() = TRUE
		&&MoreFollow= FALSE
		=>
		DomainDownload.rsp(+){
		}
		Domain_WriteBuffer()
			READY

R4	DOWNLOADING	DomainDownload.ind && Domain_DownloadSucceed() = TRUE &&MoreFollows = TRUE => DomainDownload.rsp(+){ }. Domain_WriteBuffer()	DOWNLOADING

	Current state	Event or condition => action	Next state
R5	DOWNLOADING	DomainDownload.ind && Domain_DownloadSucceed() = TRUE &&MoreFollow= FALSE => DomainDownload.rsp(+){ }. Domain_WriteBuffer()	READY

R6	DOWNLOADING	DomainDownload.ind
		&& Domain_DownloadSucceed() = FALSE
		=>
		DomainDownload.err{
		}
			DOWNLOADING

R7	DOWNLOADING	DomainDownload.ind
		&& Domain_DownloadSucceed() = FALSE
		&& DownloadFalseCounting >3
		=>
		DomainDownload.err{
		}
			EXISTENT

R8	READY	DomainDownload.ind && Domain_DownloadSucceed() = TRUE &&MoreFollow = TRUE => DomainDownload.rsp(+){ }. Domain_WriteBuffer()	DOWNLOADING
R9	READY	DomainDownload.ind && Domain_DownloadSucceed() = TRUE &&MoreFollow = FALSE => DomainDownload.rsp(+){ }. Domain_WriteBuffer()	READY

R10	READY	DownloadSequence.ind
		&& Domain_DownloadSucceed() = FALSE
		=>
		DomainDownload.err{
		}
			EXISTENT

R11	READY	IncreamentInvokeDomainCounter() && Counter=0 => Counter = 1	IN_USE
R12	IN_USE	DecreamentInvokeDomainCounter() && Counter=1 => Counter = 0	READY
R13	IN_USE	IncreamentInvokeDomainCounter() => Counter = Counter +1	IN_USE
R14	IN_USE	DecreamentInvokeDomainCounter() &&counter>1 => Counter = Counter -1	IN_USE

	Current state	Event or condition => action	Next state

R15	IN_USE	DomainDownload.ind
		=>
		DomainDownload.err{
		}
			IN_USE

R16	UPLOADING	DomainDownload.ind
		=>
		DomainDownload.rsp(-){
		ErrorType:=Service Error
		}
			UPLOADING

R17	EXISTENT	DomainUpload.ind
		=>
		DomainDownload.rsp(-){
		ErrorType:=Service Error
		}
			EXISTENT

R18	READY	DomainUpload.ind
		&&MoreFollows=TRUE
		=>
		DomainUpload.rsp(+){
		MoreFollows := TRUE
		}
			UPLOADING

R19	READY	DomainUpload.ind
		&&MoreFollows=FALSE
		=>
		DomainUpload.rsp(+){
		MoreFollows: = FALSE
		}
			READY

R20	UPLOADING	DomainUpload.ind
		&&MoreFollows=TRUE
		=>
		DomainUpload.rsp(+){
		MoreFollows = TRUE
		}
			UPLOADING

R21	UPLOADING	DomainUpload.ind
		&&MoreFollows=FALSE
		=>
		DomainUpload.rsp(+){
		MoreFollows := FALSE
		}
			READY

R22	DOWNLOADING	DomainUpload.ind
		=>
		DomainUpload.rsp(-){
		ErrorType:=Service Error
		}
			DOWNLOADING

R23	IN_USE	DomainUpload.ind
		=>
		DomainUpload.rsp (-){
		ErrorType:=Service Error
		}
			IN_USE

9.5.4.3 Functions description

Table 156 through Table 159 described the functions used in Domain ASE state transitions.

9.5.4.3.1 Domain_DownloadSucceed() description

Table 156 describes Domain_DownloadSucceed() function.

Table 156 – Domain_DownloadSucceed() description

Name	Domain_DownloadSucceed	Used in	AAE
Input		**Output**	
None		TRUE or FALSE	
Function			
Judge the download state.if failed,then return FALSE;if succeded,then return TURE.			

9.5.4.3.2 Domain_WriteBuffer() description

Table 157 describes Domain_WriteBuffer() function.

Table 157 – Domain_WriteBuffer() description

Name	Domain_WriteBuffer	Used in	AAE
Input		**Output**	
None		None	
Function			
Write the received data into buffer.			

9.5.4.3.3 IncreamentInvokeDomainCounter() description

Table 158 describes IncreamentInvokeDomainCounter() function.

Table 158 – IncreamentInvokeDomainCounter() description

Name	IncreamentInvokeDomainCounter	Used in	AAE
Input		**Output**	
None		None	
Function			
IncreamentInvokeDomainCounter incremented by 1.			

9.5.4.3.4 DecreamentInvokeDomainCounter() description

Table 159 describes DecreamentInvokeDomainCounter() function.

Table 159 – DecreamentInvokeDomainCounter() description

Name	DecreamentInvokeDomainCounter	Used in	AAE
Input		**Output**	
None		None	
Function			
DecreamentInvokeDomainCounter is decremented by 1.			

9.6 ESME Protocol State Machine

9.6.1 Primitives

9.6.1.1 The primitives and parameters exchanged between AAE and ESME

The primitives exchanged between AAE and ESME are described in clause 9.5.1.3.

The parameters used with the primitives between AAE and ESME are described in 9.5.1.4.

9.6.1.2 The primitives and parameters exchanged between SME and ESME

The primitives exchanged between SME and ESME are described in 9.4.1.3. The parameters used with the primitives exchanged between SME and ESME are described in 9.4.1.4.

9.6.1.3 Transport Layer and ESME primitive

Table 160 describes the primitives exchanged between Transport layer (UDP) and ESME.

Table 160 – The primitives exchanged between Transport Layer and ESME

Primitive name	source	Referrence parameters
EPA_PDU_req	Socket mapping entity	remote_ip_address ,data
EPA_PDU_ind	Transport layer	remote_ip_address ,data

9.6.1.4 Primitives parameters exchanged between Transport Layer and ESME

Table 161 illustrates the primitives parameters exchanged between Transport Layer and ESME.

Table 161 – Primitives parameters exchanged between Transport Layer and ESME

Parameter name	description
remote_ip_address	This parameter transfer the ip address of the remote device, namely the destination address the sending side will send to and the source address the receiving side will receive from.
data	This parameter transfer the data that the sending side will send and the receiving side will receive.

9.6.2 State description

ESME is always active. Its state is described in Table 162.

Table 162 – ESME state description

State name	description
ACTIVE	ESME transfers primitives to AAE, SME to transport layer; or it is ready to receive primitives from AAE, SME to transport layer.

9.6.3 State transitions

Figure 24 and Table 163 illustrates ESME state transitions.

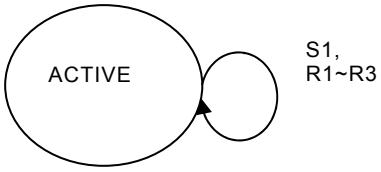


Figure 24 – ESME state transition

Table 163 – ECSME state transitions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	DTC_req
		|| DTC_rsp
		|| DTU_req
		=>
		EPA-PDU.req {
		Senddata := data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R1	ACTIVE	EPA-PDU.ind
		&& ServiceType(data) = “Confirmed Service Indication”
		=>
		DTC.ind{
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

R2	ACTIVE	EPA-PDU.ind
		&& ServiceType(data) = “Confirmed Service Confirmation”
		=>
		DTC.cnf{
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

R3	ACTIVE	EPA-PDU.ind
		&& ServiceType(data) = “Unconfirmed Service Indication”
		=>
		DTU.ind{
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

9.6.4 Function description

Table 164 describes the function used in ESME state transitions.

Table 164 – ServiceType()description

name	ServiceType	use	Socket mapping enity
input		**output**	
data		Received primitive type of service message	
function			
Judge the message type by the received service message, including confirmed service indication primitive, confirmed service primitive and unconfirmed service indication primitive.			

10 XML based EPA Device Descriptions

10.1 Overview

EPA eXtensible Device Description Language (XDDL) is designed for the interoperability of devices from different manufacturers.

When an EPA device is produced, the manufacturer should provide an EPA eXtensible Device Description (XDD) file (with the extension name .xml) to end users. The device description file shall describe network visible objects of device resources, function blocks and their parameters for the device.

This specification uses Extensible Markup Language (XML) as the device description language. Based on XML, this specification defines the necessary DD components or keywords. Vendors can use these components to write the DD files.

The DD file describes the information of EPA devices. The user can access these XDDL files by the browsers directly, or interpret these files by the DOM-based DD interpreter, in order to get the information of device function and parameter interface. A DD interpreter is used to translate the elements described in DD files into corresponding data structure.

The XML-based device description model is illustrated in Figure 25.

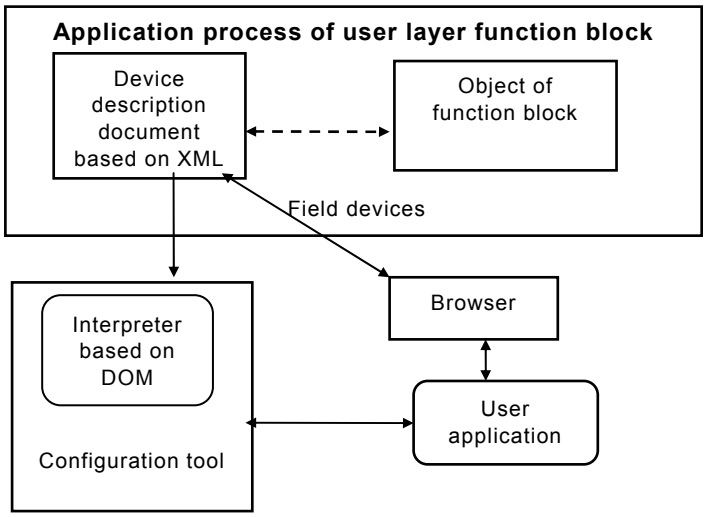


Figure 25 – DD model

10.2 The summary of EPA extensible device description file

10.2.1 Architecture

Figure 26 illustrates the structure of the XDD files:

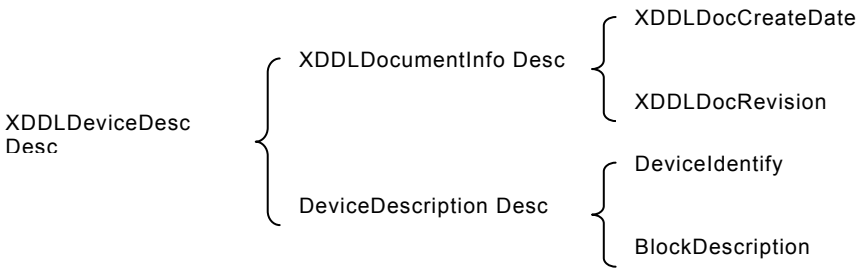


Figure 26 – Structure of XDD file

10.2.2 Overview of basic elements

This specification defines six kinds of basic elements: blocks, simple variable, structure, array, response codes and domain. Every kind of structure has some attributes, and may include a series of child elements.

a) Block

Block description specifies the external characteristics of the functional block.

b) Simple Variable

Simple Variable description specifies simple variable that is present for a host.

c) Structure

Structure description specifies structure variable that is present for user through a host.

d) Array

Array description specifies array data that is present for user through a host

e) Response codes

Response codes illustrated specific response codes for a variable, structure, array, or domain.

f) Domain

Domain description specifies the characteristics of the domain provided in an EPA device.

10.3 Structure of XDD files

10.3.1 Overview

The structure of XDD is illustrated as follows:

```
<XDDLDeviceDesc>          // start mark of root element of the device description file
<XDDLDocumentInfo>        // start mark of XDD file information description
    Information description structure
</XDDLDocumentInfo>       // end mark of XDD file information description
<DeviceDescription>       // start mark of device resource description
    Structure of device resource description
</DeviceDescription >     // end mark of device resource description
</XDDLDeviceDesc>         // end mark of root element of the device description file
```

<XDDLDeviceDesc> is the root element and start mark of an XDD file. It has no attributes. It comprises of two child elements, <XDDLDocumentInfo> and <DeviceDescription>, as shown in Table 165.

Table 165 – XDD Root element

Root element	Child element	Description
XDDLDeviceDesc	XDDLDocumentInfo	Mark the general information description of XDD file
	DeviceDescription	Mark device resource structure description

10.3.2 XDD file information structure

10.3.2.1 Overview

<XDDLDocumentInfo> marks the general information description of XDD file. The general information includes creating time and version number of the XDD file. <XDDLDocumentInfo> is composed of two child elements, <XDDLDocCreateDate> and <XDDLDocRevision>. Both of them have no attributes. Table 166 shows the structure of <XDDLDocumentInfo>.

Table 166 – Information structure element of device description file

Parent element	Child element	Descriptions
XDDLDocumentInfo	XDDLDocCreateDate	Mark the XDD file creating date description
	XDDLDocRevision	Mark the XDD file version number description

The format of date description is: yyyy-mm-dd, where yyyy represents year, mm represents month, dd represents date. One example is 2003-01-01.

The format of version number description is: MajorRevision.MinorRevision, where MajorRevision represents the major version number, MinorRevision represents minor version.

10.3.2.2 Description format

The information body of the device description file is shown as follows:

<XDDLDocCreateDate> // start mark of XDD file creating date description

yyyy-mm-dd

</XDDLDocCreateDate>

<XDDLDocRevision> // start mark of XDD file version number description

MajorRevision.MinorRevision

</XDDLDocRevision>

10.3.3 Device resource description structure

10.3.3.1 Overview

The device resource description structure describes the device characteristics and capability information. <DeviceDescription> marks device resource description. It has two child elements, <DeviceIdentify> and <BlockDescription>. <BlockDescription> has an attribute which is NumOfBlock, which describes the number of the function block in the device, as shown in Table 167:

Table 167 – Description element of device resource description structure

Parent element	Child element	Descriptions
DeviceDescription	DeviceIdentify	Mark the Device identification description
	BlockDescription	Mark function block description

10.3.3.2 Description format

An example of device resource description is shown as follows:

< DeviceIdentify > // start mark of Device identification description

Device identification description structure

</ DeviceIdentify>

<BlockDescription NumOfBlock =”numofblock”> // start mark of function block description

Functional block description structure 1

Functional block description structure 2

……

</BlockDescription>

Where, numofblock is an integer; it means the number of the functional block in the device.

10.3.3.3 Structure of Device Identification Description

10.3.3.3.1 Overview

Structure of the Device Identification Description is used to describe device manufacturer and device type. The description of structure of the Device Identification Description starts with <DeviceIdentify>, which has no attributes..

As shown in Table 168, <DeviceIdentify> has two child elements, <ManufacyureID> and <DeviceType>, both of which have no attributes.

Table 168 – Description elements for Device Identification Description Structure

Parent Element	Child Element	Element Description
DeviceIdentify	ManufactureID	describe the identification of device manufacturer
	DeviceType	describe the type of device

10.3.3.3.2 Description Format

The description format of structure of the Device Identification Description is exampled as follows:

```
<ManufactureID>                 //start mark of Device Manufacturer description
    XXXX                        // exclusive manufacturer identifier
</ManufactureID>
<DeviceType>                    // start mark of device type description
    XXXX                        //device type identification
</DeviceType>
```

10.3.3.4 Structure of Function Block Description

10.3.3.4.1 Overview

Function Block Description Structure describes all information of a function block located in a device. Function Block Description Structure description starts with <BlockDescription>, which may have one or more <Block> child elements. <Block> has two attributes, TypeID and BlockType. TypeID describes the identifier of the function block type, and BlockType describes function block type, as shown in Table 169:

Table 169 – Description elements for Function Block Description structure

Parent Element	Child Element	Descriptions
BlockDescription	BLOCK	Mark function block structure description
	…	
	BLOCK	

10.3.3.4.2 Description Format

The description of structure of Function Block Description is exampled as follows::

```
<BLOCK TypeID = "numberofBlocktype" BlockType = "BlockTypeDescription">
                                    // start mark of Function Block Description structure
Function Block Structure
</BLOCK>
```

Where, numberofBlocktype is an integer defined by manufacturer; BlockTypeDescription is a char string.

10.3.3.4.3 Function Block structure

10.3.3.4.3.1 Overview

Function Block structure describes the basis information of a function block and its all network-visible information. It starts with <BLOCK>, which has two child element, <BLOCK_INFO> and <BLOCKPARA>. <BLOCK_INFO> has an attribute, ParaType, describing type of <BLOCK_INFO>, as shown in Table 170.

Table 170 – Description elements for Function Block Structure

Parent Element	child element	Element Description
BLOCK	BLOCK_INFO	Mark basic information description of function block
	BLOCKPARA	Mark parameter description

10.3.3.4.3.2 Description Format

The description structure of the Function Block Structure is exampled as follows:

<BLOCK_INFO ParaType = "STRUCT_TYPE"> //start mark of basic information description

Basic Information Description Structure

</BLOCK_INFO>

<BLOCKPARA> // start mark of function block parameter description

Parameter Description Structure

</BLOCKPARA>

10.3.3.4.3.3 FB Basic Information Description Structure

10.3.3.4.3.3.1 Overview

Basic Information Description Structure describes the basic information of function block. It start with <BLOCK_INFOR>, which is has eleven child elements.

Table 171 – Description Elements of FB Basic Information Description Structure

Parent Element	Child Element	Element Description
BLOCK_INFO	Name	Mark FB name description
	Help	Mark help text description
	BlockTag	Mark FB tag description
	XDDLID	Mark XDDL identification description
	XDDLRevision	Mark XDDL revision number description
	Profile	Mark the FB profile description
	ProfileRevision	Mark profile revision description
	Execute Time	Mark FB execute time description
	Execute Time Unit	Mark FB execute time description
	NumOfParameters	Mark FB parameter number description
	NumOfFbInstance	Mark FB instantiation number description

10.3.3.4.3.3.2 Description Format

The description format of Structure of Function Block's Basic Information Description is as follows:

```
<Name>                          // start mark of FB name description
        FB name                 //a character string
</Name>
<Help>                          // start mark of FB help text description
        Help text                       //a character string
</Help>
<BlockTag>                      // start mark of FB default tag description
        FB default tag          // a character string
</BlockTag>
< XDDLID >                      // start mark of function block XDDL identifier
                                description
        XDDL Identifier         // a character string
</XDDLID>
<XDDLRevision>                  // start mark of function block XDDL revision number
                                description
        XDDL Revision number    // an number formatted as XX.XX
</XDDLRevision>
<Profile>                       // start mark of profile description
        Profile name                    //a character string
</Profile>
<ProfileRevision>               // start mark of revision number description
        Revision number         // an number formatted as XX.XX
</ProfileRevision>
<ExecuteTime>                   // start mark of FB execution time description
        Function block's execution time         // a positive integer
</ExecuteTime>
< ExecuteTimeUnit >             // start mark of FB execution time unit description
    time unit                   //it can be "s"--secend;"ms"--millisecond; "us"—
                                microsecond
</ExecuteTimeUnit>
<NumOfParamters>                // start mark of parameters number description
        Number of parameters of function block          //a positive integer
</NumOfParameters>
<NumOfFbInstance>               // start mark of FB instantiation number description
        FB instantiation number         //a positive integer
</NumOfFbInstance>
```

10.3.3.4.3.4 Parameter Description Structure

10.3.3.4.3.4.1 Overview

Parameter Description Structure describes all parameter element of an FB. It starts with <BLOCKPARA>, which has child element <VARIABLE>.

<VARIABLE> has an attribute, VariableType, indicating that this parameter is simple variable ,array variable, structure variable or domain variable.

10.3.3.4.3.4.2 Description Format

The description format of Parameter Description Structure is exampled as follows:

```
<VARIABLE VarialbleType = "ParaElementType">
    Parameter Element Description Structure 1
</VARIABLE>
<VARIABLE VarialbleType = "ParaElementType">
    Parameter Element Description Structure 2
</VARIABLE>
 ……
<VARIABLE VarialbleType = "ParaElementType">
    Parameter Element Description Structure N
</VARIABLE>
```

where, "ParaElementType" is the type of described parameter element. The value of it is “SIMPLE_VARIABLE” or "ARRAY_VARIABLE" or "STRUCT_VARIABLE" or “DOMAIN_VARIABLE”, indicating separately that the parameter described is simple variable, array variable, structure variable, domain variable and so on.

10.3.4 Parameter Element Description Structure

The parameter of a functional block can be one or more types among simple variable, structure, array, and domain. And every simple variable, structure, array, domain may have its own response code sets.

10.3.4.1 Simple variable element parameter Description

10.3.4.1.1 Overview

When the attribute of VarialbleType for <VARIABLE> as "SIMPLE_VARIABLE", it indicates that <VARIABLE> describes a simple variable.

As shown in Table 172, there are 16 child elements for the <VARIABLE>

Table 172 – Description elements for simple variable parameter

Parent element	Child element	Child element description
VARIABLE	Name	Mark variable name description
	Label	Mark the displayable variable name description that is offered for users (optional)
	Help	Mark help texts description (optional)
	Class	Mark input/output property description: INPUT means that this parameter is a input parameter OUTPUT means that this parameter is a output parameter CONTAINED means this parameter is an embedded parameter
	DataType	Mark data type description: BOOLEAN——Boolean ; UNSIGNED8——Unsigned8; UNSIGNED16——Unsigned16; UNSIGNED32——Unsigned32; UNSIGNED64——Unsigned64; INT8——Int8; INT16——Int16; INT32——Int32; INT64——Int64; REAL——Real; VISUALSTRING——VisualString; OCTETSTRING——OctetString; BITSTRING——BitString; TIMEOFDAY——s TimeOfDay; BINARYDATE——BinaryDate; TIMEDIFFERENCE——TimeDifference;
	Length	Mark variable length description (numbers of the byte or the bit)
	Unit	Mark variable unit description
	DefaultValue	Mark variable default value description
	MinimumValue	Mark minimum value of the range description
	MaximumValue	Mark maximum value of the range description
	UpLimit	Mark upper limited value description (optional)
	ExtraUpLimit	Mark extra upper limited value description
	DownLimit	Mark lower limited value description (optional)
	ExtraDownLimit	Mark extra lower limited value description (optional)
	ScalingFactor	Mark scale coefficient description (optional)
	OperationMode	Mark read/write property description : READ——means that the parameter can be read; WRITE——means that the parameter can be written; READ_WRITE——means that the parameter can be written and read.
	Response-codes	Mark response code description (optional)

10.3.4.1.2 Description format

The format of Simple variable element parameter Description is exampled as follows:

```
<Name>                          //start mark of variable name description
      Variable name             // a character string
</Name>
<Label>                         //start mark of displayable variable name description
      displayable variable name      // a character string
</Label>
<Help>                          //start mark of help text description
      Help text                 // a character string
</Help>
```

<Class> // start mark of description for I/O attribute
input/output attribute
</Class>
<DataType> //start mark of data type description of the variable
Data type of variable value
</DataType>
<Length> //start mark of variable length description
Length of variable
</Length>
<Unit> //start mark of variable unit description
Variable unit
</Unit>
<DefaultValue> //start mark of description for default value of a variable
default value of the variable
</DefaultValue>
<MinimumValue> //start mark of minimum value description of the variable value range
The minimum value of the variable value range
</MinimumValue>
<MaximumValue> //start mark of maximum value description of the variable value range
The maximum value of the variable value range
</MaximumValue>
<UpLimit> // start mark of upper value description
Warning value of upper limited value
</UpLimit>
<ExtraUpLimit> // start mark of extra upper value description
extra up limited value
</ExtraUpLimit>
<DownLimit> // start mark of warning value description of lower limited value
Warning value of lower limited value
</DownLimit>
<ExtraDownLimit> // start mark of warning value description of extra lower limited value
Warning value of extra lower limited value
</ExtraDownLimit>
<ScalingFactor> // start mark of scale coefficient description of the variable
Scale coefficient of the variable
</ScalingFactor>
<OperationMode> // start mark of operating description of writing and reading for the variable
The operation of writing and reading for the variable
</OperationMode>
<Response-codes> // start mark of the description of response codes
The response code
</Response-codes>

10.3.4.2 Description for array variable parameter element

10.3.4.2.1 Overview

When the attribute of VarialbleType for <VARIABLE> as " ARRAY_VARIABLE ", it indicates that <VARIABLE> describes an array variable.

The child elements of <VARIABLE> for array variable parameter description are shown in Table 173.

Table 173 – Description elements for array variable parameter

Parent element	Child element	Child element description
VARIABLE	Name	Mark variable name description
	Label	Mark the displayable variable name description that is offered for users (optional)
	Help	Mark help texts description (optional)
	Class	Mark input/output property description: INPUT means that this parameter is a input parameter OUTPUT means that this parameter is a output parameter CONTAINED means this parameter is an embedded parameter
	NumOfMembers	Mark member number description
	DataType	Mark member data type description
	Unit	Mark member unit description
	DefaultValue	Mark member default value description
	MinimumValue	Mark minimum value of member range description
	MaximumValue	Mark maximum value of member range description
	UpLimit	Mark upper limited value description (optional)
	ExtraUpLimit	Mark extra upper limited value description
	DownLimit	Mark lower limited value description (optional)
	ExtraDownLimit	Mark extra lower limited value description (optional)
	OperationMode	Mark read/write property description : READ——means that the parameter can be read; WRITE——means that the parameter can be written; READ_WRITE——means that the parameter can be written and read.

10.3.4.2.2 Description Format

The format of array variable element parameter description is exampled as follows:

```
<Name>                          //start mark of array variable name description
    Array Variable name             // a character string
</Name>
<Label>                         //start mark of displayable array variable name
                                description
    Displayable variable name       // a character string
</Label>
<Help>                          //start mark of help text description
    Help text                   // a character string
</Help>
<Class>                         // start mark of the Variable I/O property description
    The Input/Output property of the Variable
</Class>
<NumOfMembers>                  // start mark of member number description
    The number of members in the array      // a positive integer
</NumOfMembers>
<DataType>                      // start mark of Array Variable member Data Type
                                description
    Data Type of the Array Variable member
```

```
</DataType>
<Unit>                          // start mark of the Variable member Unit
        Variable member Unit
</Unit>
<DefaultValue>                  // start mark of member Default Value description
        Default Value
</DefaultValue>
<MinimumValue>                  // start mark of member Minimum Value description
        Minimum Value
</MinimumValue>
<MaximumValue>                  // start mark of member Maximum Value
        Maximum Values of Variable
</MaximumValue>
<UpLimit>                       // start mark of member Upper Limit Value description
        Upper Limit Value
</UpLimit>
<ExtraUpLimit>                  // start mark of member Extra Upper Limit Value
                                description
        Extra Upper Limit Value
</ExtraUpLimit>
<DownLimit>                     // start mark of member lower limit Value description
        lower limit Value
</DownLimit>
<ExtraDownLimit>                // start mark of member Extra lower limit
                                Valuedescription
        Extra lower limit Value
</ExtraDownLimit>
<OperationMode>                 // start mark of the read-write Operation of Array
                                Variables
        Read-write Operation of Array Variables
</OperationMode>
```

10.3.4.3 Description of structure variable parameter

10.3.4.3.1 Overview

When the attribute of VarialbleType for <VARIABLE> as " STRUCT_VARIABLE ", it indicates that <VARIABLE> describes a structure variable.

Each member of the structure variable has its own name, and each member has its own data type.

The child elements of <VARIABLE> for structure variable parameter description are shown in Table 174.

Table 174 – Description elements for structure variable parameter

Parent element	Child element	Element description
VARIABLE	Name	Mark variable name description
	Label	Mark the displayable variable name description that is offered for users (optional)
	Help	Mark help texts description (optional)
	Class	Mark input/output property description: INPUT means that this parameter is a input parameter OUTPUT means that this parameter is a output parameter CONTAINED means this parameter is an embedded parameter
	NumOfMembers	Mark structure member number description
	Members	Mark member description

<Members>, as the child element of structure variable parameter description, is used to describe all of the members in the structure, each member delegates a variable, and each of the description must follow the formation that has been used in the description of simple variable element. The description elements of structure member are determined by the detailed child parameter of the structure parameter.

Table 175 – Child elements of structure variable members description

Parent element	Child element	Descriptions
Members	Name	Mark structure member name description
	Label	Mark structure member displayable name description, optional
	Help	Mark structure member help text description
	DataType	Mark structure member data type description
	Unit	Mark structure member unit description
	DefaultValue	Mark structure member default value description
	MinimumValue	Mark structure member minimum value description
	MaximumValue	Mark structure member maximum value description
	UpLimit	Mark structure member upper limit value description
	ExtraUpLimit	Mark structure member extra upper limit value description
	DownLimit	Mark structure member lower value description
	ExtraDownLimit	Mark structure member extra lower value description
	ScalingFactor	Mark structure member scaling factor description
	OperationMode	Mark structure member read/write property description READ——The parameter is read only; WRITE——The parameter is write only; READ_WRITE——The parameter can be read and written

10.3.4.3.2 Description Format

The format of structure variable element parameter description is exampled as follows:

```
<Name>                          //start mark of structure variable name description
      Structure Variable name        // a character string
</Name>
<Label>                         //start mark of displayable structure variable name
                                description
      Displayable variable name      // a character string
</Label>
<Help>                          //start mark of help text description
      Help text                 // a character string
</Help>
<Class>                         // start mark of the Variable I/O property description
      The Input/Output property of the Variable
</Class>
<NumOfMembers>                  // start mark of member number description
      The number of members in the array     // a positive integer
</NumOfMembers>
<Members>                       // Start mark of structure member description
       <Name>                          // Start mark of the structure variable member
                                       name description
            Name of structure variable member
       </Name>
```

```
        <Label>                          // Start mark of structure variable member
                                            displayable name description
            Displayable variable name for users      // a character string
        </Label>
        <Help>                           // Start mark of help text description
            Help text for users              // a character string
        </Help>
        <DataType>                       // Start mark of structure variable member
                                            name data type description
            Data type
        </DataType>
        <Unit>                           // Start mark of structure variable member
                                            unit description
            Variable unit
        </Unit>
        <DefaultValue>                   // Start mark of structure variable member
                                            default value description
            Initial value of variable
        </DefaultValue>
        <MinimumValue>                   // Start mark of structure variable member
                                            minimum value description
            Minimum value
        </MinimumValue>
        <MaximumValue>                   // Start mark of structure variable member
                                            maximum value description
            Maximum value
        </MaximumValue>
        <UpLimit>                        // Start mark of structure variable member
                                            upper limit warning value description
             upper limit value
        </UpLimit>
        <ExtraUpLimit>                   //Start mark of structure variable member
                                            extra upper limit warning value description
            Extra upper limit value
        </ExtraUpLimit>
        <DownLimit>                      // Start mark of structure variable member
                                            lower limit value description
            lower limit value
        </DownLimit>                     // Terminal identifier for lower limit warning
                                            value
        <ExtraDownLimit>                 // Start mark of structure variable member
                                            extra lower limit value description
            Extra lower limit value
        </ExtraDownLimit>
        <ScalingFactor>                  // Start mark of structure variable member
                                            scaling factor description
            scaling description
        </ScalingFactor>
        <OperationMode>                  // Start mark of structure variable member
                                            read-write property description
            Read-write property
        </OperationMode>
</Members>
<Members>
        ……
        ……
</Members>
        ……
```

10.3.4.4 Description for enumerate variable parameter

10.3.4.4.1 Overview

When the attribute of VarialbleType for <VARIABLE> as “ENUMERATE_VARIABLE ", it indicates that <VARIABLE> describes a enumerate variable.

The child elements of <VARIABLE> for enumerate variable parameter description are shown in Table 176.

Table 176 – Description elements for enumerate variable parameter

Parent element	Child element	Element description
VARIABLE	Name	Mark enumerate variable name description
	Label	Mark enumerate variable displayable name description, optional
	Help	Mark enumerate variable help text description
	Class	Mark input/output property description: INPUT means that this parameter is a input parameter OUTPUT means that this parameter is a output parameter CONTAINED means this parameter is an embedded parameter
	DefaultValue	Mark enumerate variable default value description
	EnumerateValueList	Mark enumerate variable value list description
	OperationMode	Mark enumerate variable read/write property description READ——The parameter is read only; WRITE——The parameter is write only; READ_WRITE——The parameter can be read and written

10.3.4.4.2 Description format

The format of enumerate variable parameter description is exampled as follows:

```
<Name>                          //start mark of enumerate variable name description
        Structure Variable name         // a character string
</Name>
<Label>                         //start mark of displayable enumerate variable name
                                description
        Displayable variable name       // a character string
</Label>
<Help>                          //start mark of help text description
        Help text               // a character string
</Help>
<Class>                         // start mark of the variable I/O property description
        The Input/Output property of the Variable
</Class>
<DefaultValue>                  //start mark of enumerate variable default value
                                description
        Default value of variable
</DefaultValue>
< EnumerateValueList >          //start mark of enumerate variable value list
                                description for value list of enum variable
        Value list
</ EnumerateValueList >
```

<OperationMode> //start mark of enumerate variable operation mode description
Operation mode
</OperationMode>

10.3.4.5 Description of domain object

10.3.4.5.1 Overview

When the attribute of VarialbleType for <VARIABLE> as “DOMAIN_VARIABLE ", it indicates that <VARIABLE> describes a domain object.

The child elements of <VARIABLE> for domain object description are shown in Table 177.

Table 177 – Description elements of domain object

Parent element	Child element	Descriptions
VARIABLE	DomainName	Mark domain name description
	PhysicalAddress	Mark description for domain physical address in device
	Size	Mark description for size of domain(bytes)
	OperationMode	Mark description for download/upload operation mode of domain DOWNLOAD – the domain can be downloaded only UPLOAD – the domain can be uploaded only DOWNLOAD_UPLOAD – the domain can be either downloaded or uploaded

10.3.4.5.2 Description format

The description format of domain object element is as follows:

<DomainName> //start mark of description for domain name
Domain name
</DomainName>
< PhysicalAddress > //start mark of description for domain physical address
Domain physical address
</ PhysicalAddress >
<Size> //start mark of description for size of domain
Size of domain(bytes)
</Size>
<OperationMode> //start mark of description for operation mode of domain
Operation mode of domain
</OperationMode>

10.4 Configuration interface

XDD file can be interpreted using interfaces based on W3C Document Object Model (DOM), whose definition is independent of language and platform. It can be realized using any programming language.

Bibliography

ISO/IEC 8802-1:1996(E) ANSI/IEEE Std 802.1 *Local and metropolitan area networks – Virtual bridged local area networks*

ISO/IEC 8825-1:2002 *Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO 15745-4 *Industrial automation systems and integration — Open systems application integration framework – Part 4: Reference description for Ethernet-based control systems*

RFC 822, *Standards for ARPA Internet Text Messages*

RFC 950, *Internet Standard Subnetting Procedure*

RFC 1122, *Requirements for Internet Hosts – Communication Layers*

RFC 1155, *Structure and Identification of Management Information for TCP/IP-based Internets*

RFC 1212, *Concise MIB Definitions*

RFC 1213, *Management Information Base*

RFC 1215, *A Convention for Defining Traps for use with the SNMP*

RFC 1305, *Network Time Protocol (Version 3) Specification, Implementation and Analysis*

RFC 1643, *Definitions of Managed Objects for the Ethernet-like Interface Types*

RFC 1661-1994, *The Point -to-Point Protocol (PPP)*

Standards Survey

The IEC would like to offer you the best quality standards possible. To make sure that we continue to meet your needs, your feedback is essential. Would you please take a minute to answer the questions overleaf and fax them to us at +41 22 919 03 00 or mail them to the address below. Thank you!

Customer Service Centre (CSC)

International Electrotechnical Commission
3, rue de Varembé
1211 Genève 20
Switzerland

or

Fax to: **IEC**/CSC at +41 22 919 03 00

Thank you for your contribution to the standards-making process.

Nicht frankieren
Ne pas affranchir

Non affrancare
No stamp required

A Prioritaire

RÉPONSE PAYÉE

SUISSE

Customer Service Centre (CSC)
International Electrotechnical Commission
3, rue de Varembé
1211 GENEVA 20
Switzerland

Q1 Please report on **ONE STANDARD** and **ONE STANDARD ONLY**. Enter the exact number of the standard: *(e.g. 60601-1-1)*

.....

Q2 Please tell us in what capacity(ies) you bought the standard *(tick all that apply).* I am the/a:

purchasing agent ❑
librarian ❑
researcher ❑
design engineer ❑
safety engineer ❑
testing engineer ❑
marketing specialist ❑
other.....

Q3 I work for/in/as a: *(tick all that apply)*

manufacturing ❑
consultant ❑
government ❑
test/certification facility ❑
public utility ❑
education ❑
military ❑
other.....

Q4 This standard will be used for: *(tick all that apply)*

general reference ❑
product research ❑
product design/development ❑
specifications ❑
tenders ❑
quality assessment ❑
certification ❑
technical documentation ❑
thesis ❑
manufacturing ❑
other.....

Q5 This standard meets my needs: *(tick one)*

not at all ❑
nearly ❑
fairly well ❑
exactly ❑

Q6 If you ticked NOT AT ALL in Question 5 the reason is: *(tick all that apply)*

standard is out of date ❑
standard is incomplete ❑
standard is too academic ❑
standard is too superficial ❑
title is misleading ❑
I made the wrong choice ❑
other .....

Q7 Please assess the standard in the following categories, using the numbers:
(1) unacceptable,
(2) below average,
(3) average,
(4) above average,
(5) exceptional,
(6) not applicable

timeliness.....
quality of writing.....
technical contents.....
logic of arrangement of contents .....
tables, charts, graphs, figures.....
other .....

Q8 I read/use the: *(tick one)*

French text only ❑
English text only ❑
both English and French texts ❑

Q9 Please share any comment on any aspect of the IEC that you would like us to know:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ISBN 2-8318-8081-5

ICS 25.040.40; 35.240.50; 35.100.05

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND