IEC 61158-6-14

Edition 3.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

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

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-14: Spécification du protocole de la couche application – Eléments de type 14

IEC 61158-6-14:2014-08(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

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

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

IEC Catalogue - webstore.iec.ch/catalogue
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

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 It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

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

A propos de l'IEC
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue
Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,…). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 14 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary
Plus de 55 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC 61158-6-14

Edition 3.0 2014-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

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

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-14: Spécification du protocole de la couche application – Eléments de type 14

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XD
CODE PRIX

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1764-1

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

FOREWORD

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

Attention is drawn to the fact that the use of the associated protocol type is restricted by its intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a layer protocol type to be used with other layer protocols of the same type, or in other type combinations explicitly authorized by its intellectual-property-right holders.

NOTE Combinations of protocol types are specified in IEC 61784-1 and IEC 61784-2.

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

This third edition cancels and replaces the second edition published in 2010. This edition constitutes a technical revision. The main changes with respect to the previous edition are listed below:

- corrections of editorial errors;
- specification changes for CPF4;
- update of the requirements for all conformance classes;
- update of the requirements for all conformance services.

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

FDIS	Report on voting
65C/764/FDIS	65C/774/RVD

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

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

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

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

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

INTRODUCTION

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

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

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

1 Scope

1.1 General

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

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 14 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard specifies interactions between remote applications and defines the externally visible behavior provided by the Type 14 fieldbus application layer in terms of

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

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

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

This standard specifies the protocol of the Type 14 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-14.

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

1.3 Conformance

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

2 Normative references

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

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

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

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

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

IEC 61158-6 (all parts), *Industrial communication networks – Fieldbus specifications – Part 6: Application layer protocol specification*

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

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

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

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

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

ISO/IEC 8824:1990, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*[1]

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

[1] Withdrawn.

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

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

IEEE 754-2008, *IEEE Standard for Floating-Point Arithmetic*

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

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

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

3.1.2 ISO/IEC 8822 terms

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

a) abstract syntax
b) presentation context

3.1.3 ISO/IEC 9545 terms

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

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

3.1.4 ISO/IEC 8824 terms

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

a) object identifier

b) type

3.1.5 Fieldbus data-link Layer terms

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

a) DL-Time

b) DL-Scheduling-policy

c) DLCEP

d) DLC

e) DL-connection-oriented mode

f) DLPDU

g) DLSDU

h) DLSAP

i) link

j) network address

k) node address

l) node

m) scheduled

3.2 Fieldbus application layer specific terms and definitions

3.2.1
access control
control on the reading and writing of an object

3.2.2
access path
association of a symbolic name with a variable for the purpose of open communication

3.2.3
communication macrocycle
set of basic cycles needed for a configured communication activity in a macro network segment

3.2.4
communication scheduling
algorithms and operation for data transfers occurring in a deterministic and repeatable manner

3.2.5
configuration (of a system or device)
step in system design: selecting functional units, assigning their locations and defining their interconnections

3.2.6
cyclic
repetitive in a regular manner

3.2.7
destination FB Instance
FB instance that receives the specified parameters

3.2.8
domain
part of memory used to store code or data

3.2.9
domain download
operation to write data in a domain

3.2.10
domain upload
operation to read data from a domain

3.2.11
entity
particular thing, such as a person, place, process, object, concept, association, or event

3.2.12
Type 14 bridge
DL-relay entity which performs synchronization between links (buses) and may perform selective store-and-forward and routing functions to connect two micro network segments

3.2.13
identifier
16-bit word associated with a system variable

3.2.14
index
address of an object within an application process

3.2.15
instance
actual physical occurrence of an object within a class that identifies one of many objects within the same object class

3.2.16
instantiation
creation of an instance of a specified type

3.2.17
management information
network-visible information for the purpose of managing the field system

3.2.18
management information base
organized list of management information

3.2.19
mapping
set of values having defined correspondence with the quantities or values of another set

3.2.20
member
piece of an attribute that is structured as an element of an array

3.2.21
message filtering
decision on a message according to a special rule

3.2.22
micro segment
part of a network, where special scheduling is implemented

3.2.23
offset
number of octets from a specially designated position

3.2.24
phase
elapsed fraction of a cycle, measured from some fixed origin

3.2.25
process interface
data exchange and information mapping between physical process and application unit

3.2.26
real-time
ability of a system to provide a required result in a bounded time

3.2.27
real-time communication
transfer of data in real-time

3.2.28
Real-Time Ethernet
RTE
ISO/IEC 8802-3-based network that includes real-time communication

Note 1 to entry: Other communication can be supported, providing the real-time communication is not compromised.

Note 2 to entry: This definition is dedicated, but not limited, to ISO/IEC 8802-3. It could be applicable to other IEEE 802 specifications, for example IEEE 802.1Q.

3.2.29
schedule
temporal arrangement of a number of related operations

3.2.30
scheduling macrocycle
time interval to implement a specific schedule

3.2.31
source FB Instance
FB instance that sends a specific parameter

3.2.32
time offset
time difference from a specially designated time

3.3 Abbreviations and symbols

AAE	Application Access Entity
AE	Application Entity
AL	Application Layer
ALE	Application Layer Entity
ALP	Application Layer Protocol

APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
ARP	Address Resolution Protocol
AREP	Application Relationship End Point
ASE	Application Service Element
Cnf	Confirmation
CR	Communication Relationship
CREP	Communication Relationship End Point
CSMA/CD	Carrier Sense Multiple Access Protocol with Collision Detection
DD	Device Description
DHCP	Dynamic Host Configuration Protocol
DL-	(as a prefix) data-link-
DLCEP	Data-link Connection End Point
DLL	Data-link Layer
DLE	Data-link Entity
DLM	Data-link-management
DLS	Data-link Service
DLSAP	Data-link Service Access Point
DLSDU	DL-service-data-unit
ECSME	Type 14 communication scheduling management entity
EM_	(as a prefix) Type 14 Management
ESME	Type 14 Socket Mapping Entity
FB	Function Block
FBAP	Function Block Application Process
FME	FAL Management Entity
FRT	Fast Real-time
Ind	Indication
IP	Internet Protocol
LLC	Logical Link Control
LMP	Link Management Protocol
MAC	Medium Access Control
MAU	Medium Attachment Unit
MOB	Management Object Base
PAD	Pad (bits)
PDU	Protocol Data Unit
P/S	Publisher/Subscriber
Req	Request
Rsp	Response
RTE	Real-Time Ethernet
RT-Ethernet	Real-Time Ethernet
SAP	Service Access Point
SDU	Service Data Unit
SME	System Management Entity

SNTP	Simple Network Time Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
.cnf	Confirm Primitive
.ind	Indication Primitive
.req	Request Primitive
.rsp	Response Primitive

3.4 Conventions

3.4.1 General concept

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

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in IEC 61158-5-14. The service specification defines the services that are provided by the ASE.

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

3.4.2 Conventions for state machines for Type 14

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

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

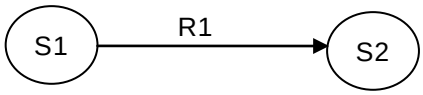


Figure 1 – State transition diagram

Table 1 – State machine description elements

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

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

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

xxx A parameter name.

Example:

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

Example:

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

= A logical condition to indicate an item on the left is equal to an item on the right.
< A logical condition to indicate an item on the left is less than the item on the right.
> A logical condition to indicate an item on the left is greater than the item on the right.
<> A logical condition to indicate an item on the left is not equal to an item on the right.
&& Logical "AND"
|| Logical "OR"

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

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

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

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

4 Abstract syntax

4.1 Fixed format PDU description

Type 14 PDU consists of fixed-length PDU header and variable-length PDU body. The former contains service type, message type and message length, etc.

```
Type 14 PDU : : = CHOICE {
        confirmed-RequestPDU        [0]   IMPLICIT  Confirmed-RequestPDU,
        confirmed-ResponsePDU       [1]   IMPLICIT  Confirmed-ResponsePDU,
        confirmed-ErrorPDU          [2]   IMPLICIT  Confirmed-ErrorPDU,
        unconfirmed-RequestPDU      [3]   IMPLICIT  Unconfirmed-RequestPDU
}
Confirmed-RequestPDU : : = SEQUENCE {
        pduHeader                   PDUHeader,
        confirmed-request           Confirmed-Request
}
Confirmed-ResponsePDU : : = SEQUENCE {
        pduHeader                   PDUHeader,
        confirmed-response          Confirmed-Response
}
Confirmed-ErrorPDU : : = SEQUENCE {
        pduHeader                   PDUHeader,
        confirmed-error             Confirmed-Error
}
Unconfirmed-RequestPDU : : = SEQUENCE {
        pduHeader                   PDUHeader,
        unconfirmed-request         Unconfirmed-Request
}
```

4.1.1 Confirmed request service

```
Confirmed- Request : : = CHOICE {
```

```
        EM_GetDeviceAttribute      [0]     IMPLICIT  EM_GetDeviceAttribute-RequestPDU,
        EM_ConfiguringDevice       [1]     IMPLICIT  EM_ConfiguringDevice-RequestPDU,
        EM_SetDefaultValue         [2]     IMPLICIT  EM_SetDefaultValue-RequestPDU,
        DomainDownload             [3]     IMPLICIT  DomainDownload-RequestPDU,
        DomainUpload               [4]     IMPLICIT  DomainUpload-RequestPDU,
        AcknowledgeEventRoport     [5]     IMPLICIT  AcknowledgeEventNotifi-RequestPDU,
        ReportConditionChanging    [6]     IMPLICIT  AlterEventConditionMon-RequestPDU,
        Read                       [7]     IMPLICIT  Read-RequestPDU,
        Write                      [8]     IMPLICIT  Write-RequesPDU,
        FRTRead                    [9]     IMPLICIT  FRTRead-RequestPDU,
        FRTWrite                   [10]    IMPLICIT  FRTWrite-RequesPDU
        BlockTransmissionOpen      [11]    IMPLICIT  OpenBlockTransmission-RequestPDU,
        BlockTransmissionClose     [12]    IMPLICIT  CloseBlockTransmission-RequestPDU,
}
```

4.1.2 Confirmed response service

```
Confirmed- Response : : = CHOICE {
        EM_GetDeviceAttribute      [0]  IMPLICIT  EM_GetDeviceAttribute-ResponsePDU,
        EM_ConfiguringDevice       [1]  IMPLICIT  EM_ConfiguringDevice-ResponsePDU,
        EM_SetDefaultValue         [2]  IMPLICIT  EM_SetDefaultValue-ResponsePDU,
        DomainDownload             [3]  IMPLICIT  DomainDownload-ResponsePDU,
        DomainUpload               [4]  IMPLICIT  DomainUpload-ResponsePDU,
        AcknowledgeEventRoport     [5]  IMPLICIT  AcknowledgeEventNotifi-ResponsePDU,
        ReportConditionChanging    [6]  IMPLICIT  AlterEventConditionMon-ResponsePDU,
        Read                       [7]  IMPLICIT  Read-ResponsePDU,
        Write                      [8]  IMPLICIT  Write-ResponsePDU,
        FRTRead                    [9]  IMPLICIT  FRTRead-ResponsePDU,
        FRTWrite                   [10] IMPLICIT  FRTWrite-ResponsePDU
        BlockTransmissionOpen      [11] IMPLICIT  OpenBlockTransmission-ResponsePDU,
        BlockTransmissionClose     [12] IMPLICIT  CloseBlockTransmission-ResponsePDU,
}
```

4.1.3 Confirmed error

```
Confirmed- Error : : = CHOICE {
        EM_GetDeviceAttribute      [0]     IMPLICIT  Error-Type,
        EM_ConfiguringDevice       [1]     IMPLICIT  Error-Type,
        EM_SetDefaultValue         [2]     IMPLICIT  Error-Type,
        DomainDownload             [3]     IMPLICIT  Error-Type,
        DomainUpload               [4]     IMPLICIT  Error-Type,
        AcknowledgeEventRoport     [5]     IMPLICIT  Error-Type,
        ReportConditionChanging    [6]     IMPLICIT  Error-Type,
        Read                       [7]     IMPLICIT  Error-Type,
        Write                      [8]     IMPLICIT  Error-Type,
        FRTRead                    [9]     IMPLICIT  Error-Type,
        FRTWrite                   [10]    IMPLICIT  Error-Type
        BlockTransmissionOpen      [11]    IMPLICIT  Error-Type
        BlockTransmissionClose     [12]    IMPLICIT  Error-Type
}
```

4.1.4 Error type

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

4.1.5 Error class

```
ErrorClass ::=  CHOICE {                                                    ErrorCode
    Resource                       [0]   IMPLICIT  Integer8 {
                                             memory-unavailable             (0),
```

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

4.1.6 Unconfirmed request

```
Unconfirmed-Request: : = CHOICE {
        EM_DetectingDevice          [0]     IMPLICIT  EM_DetectingDevice-RequestPDU,
        EM_OnlineReply              [1]     IMPLICIT  EM_OnlineReply-RequestPDU,
        EM_ActiveNotification       [2]     IMPLICIT  EM_ActiveNotification-RequestPDU,
        EventRoport                 [3]     IMPLICIT  EventRoport-RequestPDU,
        VariableDistribute          [4]     IMPLICIT  VariableDistribute-RequestPDU,
        FRTVariableDistribute       [5]     IMPLICIT  FRTVariableDistribute-RequestPDU
        BlockTransmit               [6]     IMPLICIT  BlockTransmit-RequestPDU
        BlockTransmissionHeartbeat[7]       IMPLICIT  RequestPDU-RequestPDU
}
```

4.1.7 Type 14 application layer PDU

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

4.1.8 APDU header format

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

4.1.9 FAL Management Entity services

4.1.9.1 EM_DetectingDevice service

```
EM_DetectingDevice-RequestPDU : : = SEQUENCE {
        QueryType                   [0]     IMPLICIT  Unsigned8,
```

```
        Reserved                    [1]    IMPLICIT  OctetString,
        PDTag                       [2]    IMPLICIT  VisibleString,
        FBTag                       [3]    IMPLICIT  VisibleString,
        ElementID                   [4]    IMPLICIT  Unsigned16
}
```

4.1.9.2 EM_OnlineReply service

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

4.1.9.3 EM_GetDeviceAttribute service

```
EM_GetDeviceAttribute-RequestPDU : : = SEQUENCE {
        DestinationIPAddress        [0]    IMPLICIT  Unsigned32,
}
EM_GetDeviceAttribute-ResponsePDU : : = CHOICE {
        EM_GetDeviceAttribute-PositiveResponsePDU,
        EM_GetDeviceAttribute-NegativeResponsePDU
}
EM_GetDeviceAttribute-PositiveResponsePDU : : = SEQUENCE {
        DeviceID                    [0]    IMPLICIT  VisibleString,
        PdTag                       [1]    IMPLICIT  VisibleString,
        Status                      [2]    IMPLICIT  Unsigned8,
        DeviceType                  [3]    IMPLICIT  Unsigned8,
        Annunciation Interval       [4]    IMPLICIT  Unsigned16,
        Annunciation Version Number [5]     IMPLICIT  Unsigned16,
        DuplicateTagDetected        [6]    IMPLICIT  Boolean,
        DeviceRedundancyNumber      [7]    IMPLICIT  Unsigned8,
        LANRedundancyPort           [8]    IMPLICIT  Unsigned16
        DeviceRedundancy State      [9]    IMPLICIT  Unsigned8,
        MaxRedundancyNumber         [10]   IMPLICIT  Unsigned8,
        ActiveIPAddress             [11]   IMPLICIT  Unsigned32
}
EM_GetDeviceAttribute-NegativeResponsePDU : : = SEQUENCE {
        DestinationIPAddress        [0]    IMPLICIT      Unsigned32,
        Error-Type                  [1]    IMPLICIT      Error-Type
}
```

4.1.9.4 EM_ActiveNotification service

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

4.1.9.5 EM_ConfiguringDevice service

```
EM_ConfiguringDevice-RequestPDU : : = SEQUENCE {
        DestinationIPAddress        [0]    IMPLICIT  Unsigned32,
```

```
        DeviceID                    [1]   IMPLICIT  VisibleString,
        PdTag                       [2]   IMPLICIT  VisibleString,
        AnnunciationInterval        [3]   IMPLICIT  Unsigned16,
        DuplicateTagDetected        [4]   IMPLICIT  Boolean,
        DeviceRedundancyNumber      [5]   IMPLICIT  Unsigned8,
        LANRedunancyPort            [6]   IMPLICIT  Unsigned16,
        DeviceRedundancyState       [7]   IMPLICIT  Unsigned8,
        MaxRedundancyNumber         [8]   IMPLICIT  Unsigned8,
        ActiveIPAddress             [9]   IMPLICIT  Unsigned32
}
EM_ConfiguringDevice-ResponsePDU : : = CHOICE {
        EM_ConfiguringDevice-PositiveResponsePDU,
        EM_ConfiguringDevice-NegativeResponsePDU
}
EM_ConfiguringDevice-PositiveResponsePDU : : = SEQUENCE {
        DestinationIPAddres         [0]   IMPLICIT  Unsigned32,
        MaxRedundancyNumber         [1]   IMPLICIT  Unsigned8
}
EM_ConfiguringDevice-NegativeResponsePDU : : = SEQUENCE {
        DestinationIPAddress        [0]   IMPLICIT  Unsigned32,
        ErrorType                   [1]   IMPLICIT  ErrorType
}
```

4.1.9.6 EM_SetDefaultValue service

```
EM_SetDefaultValue-RequestPDU : : = SEQUENCE {
        DestinationIPAddress        [0]   IMPLICIT  Unsigned32,
        DeviceID                    [1]   IMPLICIT  VisibleString,
        PdTag                       [2]   IMPLICIT  VisibleString
}
EM_SetDefaultValue-ResponsePDU : : = CHOICE {
        EM_SetDefaultValue-PositiveResponsePDU ,
        EM_SetDefaultValue-NegativeResponsePDU
}
EM_SetDefaultValue-PositiveResponsePDU : : = SEQUENCE {
        DestinationIPAddress        [0]   IMPLICIT  Unsigned32
}
EM_SetDefaultValue-NegativeResponsePDU : : = SEQUENCE {
        DestinationIPAddress        [0]   IMPLICIT  Unsigned32,
        ErrorType                   [1]   IMPLICIT  ErrorType
}
```

4.1.10 Application Access Entity (AAE) services

4.1.10.1 DomainDownload service

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

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

4.1.10.2 DomainUpload service

```
DomainUpload-RequestPDU : : = SEQUENCE {
        SourceAppID             [0]   IMPLICIT  Unsigned16,
        DestinationAppID        [1]   IMPLICIT  Unsigned16,
        DestinationObjectID     [2]   IMPLICIT  Unsigned16,
        DataNumber              [3]   IMPLICIT  Unsigned16
}
DomainUpload-ResponsePDU : : = CHOICE {
        DomainUpload-PositiveResponsePDU,
        DomainUpload-NegativeResponsePDU
}
DomainUpload-PositiveResponsePDU : : = SEQUENCE {
        DestinationAppID        [0]   IMPLICIT  Unsigned16,
        DataLength              [1]   IMPLICIT  Unsigned16,
        MoreFollows             [2]   IMPLICIT  Boolean,
        Reserved                [3]   IMPLICIT  OctetString,
        LoadData                [4]   IMPLICIT  OctetString
}
DomainUpload-NegativeResponsePDU : : = SEQUENCE {
        DestinationAppID        [0]     IMPLICIT  Unsigned16,
        Reserved                [1]     IMPLICIT  OctetString,
        ErrorType               [2]     IMPLICIT  ErrorType
}
```

4.1.10.3 EventRoport service

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

4.1.10.4 AcknowledgeEventRoport service

```
AcknowledgeEventRoport-RequestPDU : : = SEQUENCE {
        DestinationAppID        [0]   IMPLICIT  Unsigned16,
        DestinationObjectID     [1]   IMPLICIT  Unsigned16,
        EventNumber             [2]   IMPLICIT  Unsigned16
}
AcknowledgeEventRoport -ResponsePDU : : = CHOICE {
        AcknowledgeEventRoport -PositiveResponsePDU,
        AcknowledgeEventRoport -NegativeResponsePDU
}
AcknowledgeEventRoport -PositiveResponsePDU: : = SEQUENCE{
        DestinationAppID        [0]   IMPLICIT  Unsigned16
}
AcknowledgeEventRoport -NegativeResponsePDU: : = SEQUENCE{
        DestinationAppID        [0]   IMPLICIT  Unsigned16
        Reserved                [1]   IMPLICIT  OctetString,
        ErrorType               [2]   IMPLICIT  ErrorType
}
```

4.1.10.5 ReportConditionChanging service

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

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

4.1.10.6 Read service

```
Read-RequestPDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
        DestinationObjectID         [1]   IMPLICIT  Unsigned16,
        SubIndex                    [2]   IMPLICIT  Unsigned16
}
Read -ResponsePDU : : = CHOICE {
        Read -PositiveResponsePDU,
        Read -NegativeResponsePDU
}
Read-PositiveResponsePDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
        Reserved                    [1]   IMPLICIT  OctetString,
        Data                        [2]   IMPLICIT  OctetString
}
Read-NegativeResponsePDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
        Reserved                    [1]   IMPLICIT  OctetString,
        ErrorType                   [2]   IMPLICIT  ErrorType
}
```

4.1.10.7 Write service

```
Write-RequestPDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
        DestinationObjectID         [1]   IMPLICIT  Unsigned16,
        SubIndex                    [2]   IMPLICIT  Unsigned16,
        Reserved                    [3]   IMPLICIT  OctetString,
        Data                        [4]   IMPLICIT  OctetString
}
Write -ResponsePDU : : = CHOICE {
        Write -PositiveResponsePDU,
        Write -NegativeResponsePDU
}
Write -PositiveResponsePDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
}
Write -NegativeResponsePDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
        Reserved                    [1]   IMPLICIT  OctetString,
        ErrorType                   [2]   IMPLICIT  ErrorType
}
```

4.1.10.8 VariableDistribute service

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

4.1.10.9 FRTRead service

```
FRTRead-RequestPDU : : = SEQUENCE {
        DestinationObjectID         [0]   IMPLICIT  Unsigned16,
        SubIndex                    [1]   IMPLICIT  Unsigned16
}
FRTRead -ResponsePDU : : = CHOICE {
        FRTRead -PositiveResponsePDU,
        FRTRead -NegativeResponsePDU
}
FRTRead-PositiveResponsePDU : : = SEQUENCE {
        FRTData                     [0]   IMPLICIT  OctetString
}
FRTRead-NegativeResponsePDU : : = SEQUENCE {
        ErrorType                   [0]   IMPLICIT  ErrorType
}
```

4.1.10.10 FRTWrite service

```
FRTWrite-RequestPDU : : = SEQUENCE {
        DestinationObjectID         [0]   IMPLICIT  Unsigned16,
        SubIndex                    [1]   IMPLICIT  Unsigned16,
        Reserved                    [2]   IMPLICIT  OctetString,
        Data                        [3]   IMPLICIT  OctetString
}
FRTWrite -ResponsePDU : : = CHOICE {
        FRTWrite -PositiveResponsePDU,
        FRTWrite -NegativeResponsePDU
}
FRTWrite -PositiveResponsePDU : : = SEQUENCE {
        DestinationObjectID         [0]   IMPLICIT  Unsigned16,
}
FRTWrite -NegativeResponsePDU : : = SEQUENCE {
        ErrorType                   [0]   IMPLICIT  ErrorType
}
```

4.1.10.11 FRTVariableDistribute service

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

4.1.10.12 BlockTransmissionOpen service

```
BlockTransmissionOpen-RequestPDU : : = SEQUENCE {
        SourceAppID                 [0]   IMPLICIT  Unsigned16,
        DestinationAppID            [1]   IMPLICIT  Unsigned16
        DestinationObjectID         [2]   IMPLICIT  Unsigned16
        BlockType                   [3]   IMPLICIT  Unsigned16
        BlockConfigInfo             [4]   IMPLICIT  OctetString
}
BlockTransmissionOpen-ResponsePDU : : = CHOICE {
        BlockTransmissionOpen -PositiveResponsePDU,
        BlockTransmissionOpen -NegativeResponsePDU
}
BlockTransmissionOpen-PositiveResponsePDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
        Reserved                    [1]   IMPLICIT  OctetString,
        MultiIPAddress              [2]   IMPLICIT  Unsigned32
}
BlockTransmissionOpen-NegativeResponsePDU : : = SEQUENCE {
        DestinationAppID            [0]   IMPLICIT  Unsigned16,
        Reserved                    [1]   IMPLICIT  OctetString,
        ErrorType                   [2]   IMPLICIT  ErrorType
```

```
}
```

4.1.10.13 BlockTransmissionClose service

```
BlockTransmissionClose-RequestPDU : : = SEQUENCE {
        SourceAppID              [0]   IMPLICIT  Unsigned16,
        DestinationAppID         [1]   IMPLICIT  Unsigned16
        DestinationObjectID      [2]   IMPLICIT  Unsigned16
        BlockType                [3]   IMPLICIT  Unsigned16
}
BlockTransmissionClose-ResponsePDU : : = CHOICE {
        BlockTransmissionClose -PositiveResponsePDU,
        BlockTransmissionClose -NegativeResponsePDU
}
BlockTransmissionClose-PositiveResponsePDU : : = SEQUENCE {
        DestinationAppID         [0]   IMPLICIT  Unsigned16,
}
BlockTransmissionClose-NegativeResponsePDU : : = SEQUENCE {
        DestinationAppID         [0]   IMPLICIT  Unsigned16,
        Reserved                 [1]   IMPLICIT  OctetString,
        ErrorType                [2]   IMPLICIT  ErrorType
}
```

4.1.10.14 BlockTransmit service

```
BlockTransmit-RequestPDU : : = SEQUENCE {
        SourceAppID              [0]   IMPLICIT  Unsigned16,
        DestinationAppID         [1]   IMPLICIT  Unsigned16,
        DestinationObjectID      [2]   IMPLICIT  Unsigned16,
        DataLength               [3]   IMPLICIT  Unsigned16,
        BlockType                [4]   IMPLICIT  Unsigned16
        SequenceNumber           [5]   IMPLICIT  Unsigned16
        TimeStamp                [6]   IMPLICIT  PrecisionTimeDifference
        SendCount                [7]   IMPLICIT  Unsigned16
        BlockData                [8]   IMPLICIT  OctetString
}
```

4.1.10.15 BlockTransmissionHeartbeat service

```
BlockTransmissionHeartbeat-RequestPDU : : = SEQUENCE {
        SourceAppID              [0]  IMPLICIT  Unsigned16,
        DestinationAppID         [1]   IMPLICIT  Unsigned16,
        DestinationObjectID      [2]   IMPLICIT  Unsigned16,
        ReceptionCount           [3]   IMPLICIT  Unsigned16,
        CumulativeLost           [4]   IMPLICIT  Unsigned16
        Jitter                   [5]   IMPLICIT  PrecisionTimeDifference
}
```

4.1.11 Abstract syntax of data type

4.1.11.1 Notation of Boolean type

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

4.1.11.2 Notation of integer type

```
Int8 ::= INTEGER  (-128..+127)       -- integer range -27<=  i  <=  27-1
Int16 ::= INTEGER (-32 768..+32 767) -- integer range -215<=  i  <=  215-1
Int32 ::= INTEGER                    -- integer range -231<=  i  <=  231-1
Int64 ::= INTEGER                    -- integer range -263<=  i  <=  263-1
```

4.1.11.3 Notation of unsigned integer type

```
Unsigned8 ::= INTEGER  (0..255)      -- integer range 0 <=  i  <=  28-1
Unsigned16 ::= INTEGER (0..65 535)   -- integer range 0 <=  i  <=  216-1
Unsigned32 ::= INTEGER               -- integer range 0 <=  i  <=  232-1
Unsigned64 ::= INTEGER               -- integer range 0 <=  i  <=  264-1
```

4.1.11.4 Notation of float data type

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

4.1.11.5 Notation of visible string type

VisibleString ::= VISIBLE STRING --general use

4.1.11.6 Notation of octet string type

OctetString ::= Octet STRING --general use

4.1.11.7 Notation of bit string type

BitString ::= BIT STRING -- general use

4.1.11.8 Notation of TimeofDay type

TimeOfDay ::= OctetString6

4.1.11.9 Notation of binary date type

BinaryDate ::= OctetString8

4.2 Object definitions in FAL management ASE

4.2.1 Type 14 MOB Header

The object of the Type 14 MOB Header is defined as shown in Table 2.

Table 2 – Definition of Type 14 MOB header object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	the index of Type 14 MOB header object in the Type 14 MOB
2	MOB Revision Number	Read Only	Unsigned16	2	2	The version of Type 14 MOB

4.2.2 Type 14 device descriptor object

The object of the Type 14 Device Descriptor Object is defined as shown in Table 3.

Table 3 – Definition of Type 14 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 Type 14 Device Descriptor Object in the MOB
2	Reserved	Read Only	Unsigned8	2	1	reserved
3	Application Type	Read Only	Unsigned8	3	1	application type
4	Device ID	Read Only	VisibleString	4	32	device ID
5	PD_Tag	Read Only	VisibleString	36	32	device Tag
6	Active IP Address	Read Only	Unsigned32	68	4	current operational IP address
7	Device Type	Read Only	Unsigned8	72	1	device type
8	Status	Read Only	Unsigned8	73	1	device status
9	Device Version	Read Only	Unsigned16	74	2	device version number
10	Annunciation Interval	Read Only	Unsigned16	76	2	the interval of devices broadcast its annunciation
11	Annunciation Version Number	Read Only	Unsigned16	78	2	annunciation version number of devices broadcast

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
12	Device Redundancy State	Read Only	Unsigned8	80	1	device redundancy status
13	Device Redundancy Number	Read Only	Unsigned8	81	1	device redundancy number
14	LANRedundancyPort	Read Only	Unsigned16	82	2	redundant messages processing port of the device
15	Max Redundancy Number	Read Only	Unsigned8	84	1	maximum redundancy number of the device
16	Duplicate Tag Detected	Read Only	Boolean	85	1	this property describes whether the device's PD_Tag is in collision with another device

4.2.3 Time synchronization object

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

Table 4 – Definition of the time synchronization object

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

4.2.4 Maximum response time object

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

Table 5 – Definition of maximum response time object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	the index of object in the MOB
2	Reserved	Read Only	OctetString	2	2	reserved

3	Max Response Time	Read/Write	PrecisionTimeDifference	4	8	the maximum response time of confirmed service in nanoseconds

4.2.5 Type 14 communication scheduling management object

The object of the Type 14 Communication Scheduling Management class is defined as shown in Table 6.

Table 6 – Definition of the Type 14 communication scheduling management object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	the index of object in the MOB
2	Reserved	Read Only	OctetString	2	2	reserved
3	Communication MacroCycle	Read/Write	PrecisionTimeDifference	4	8	the communication macro period of subnet which the device belongs to. The unit is nanoseconds
4	NonPeriodic Data Transfer Offset	Read/Write	PrecisionTimeDifference	12	8	the offset of non-periodic message begin to transmit relative to the start of communication macro period. The unit is nanoseconds
5	Communication Macrocycle Version Number	Read Only	Unsigned16	20	2	the version number of communication macro period

4.2.6 Device application information object

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

Table 7 – Definition of the device application information object

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

4.2.7 FB application information header

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

Table 8 – Definition of FB application information header

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	the index of object in the MOB
2	Number of FB Application Information Object	Read Only	Unsigned16	2	2	the number of FB Application Information Object
3	First Number of FB Application Information Object	Read Only	Unsigned16	4	2	first Number of FB Application Information Object

4.2.8 Domain application information header

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

Table 9 – Definition of domain application information header

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

4.2.9 Type 14 link object header

The object of the Type 14 Link Object Header class is defined as shown in Table 10.

Table 10 – Definition of Type 14 link object header

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

4.2.10 Type 14 link object header

The object of the Type 14 Link Object Header class is defined as shown in Table 11.

Table 11 – Definition of Type 14 FRT 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 FRT Link Object Header t in the MOB
2	Number of FRT Link Object	Read Only	Unsigned16	2	2	number of FRT Link Object in the device
3	First Number of FRT Link Object	Read Only	Unsigned16	4	2	first Number of FRT Link Object in the MOB
4	Number of Configured FRT Link Object	Read Only	Unsigned16	6	2	number of Configured FRT Link Objects
5	Number of UnConfigured FRT Link Object	Read Only	Unsigned16	8	2	number of UnConfigured FRT Link Objects

4.2.11 FB application information object

The object of the FB Application Information class is defined as shown in Table 12.

Table 12 – Definition of FB application information object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	the index of FB Application Information Object in the MOB
2	Reserved	Read Only	Unsigned16	2	2	reserved
3	FB Name	Read Only	VisibleString	4	32	FB Name, its length is 32 octets, if the string length is less than 32 octets, then the remained part are padded with BLANK(0x20)
4	FB Type	Read Only	Unsigned16	36	2	FB Type
5	Max Number of Instantiation	Read Only	Unsigned16	38	2	the maximum instances number of FB
6	FB Execution Time	Read Only	Unsigned32	40	4	the execution time of FB in milliseconds
7	First Number of Instantiation	Read Only	Unsigned16	44	2	first Instance Number allocated to FB instance when it is instantiated

4.2.12 Type 14 link object

The object of the Type 14 Link Object class is defined as shown in Table 13.

Table 13 – Definition of Type 14 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 Type 14 Link Object in the MOB
2	LocalAppID	Read/Write	Unsigned16	2	2	instance ID of local instance
3	Local Object ID	Read/Write	Unsigned16	4	2	the index of local variable object
4	RemoteAppID	Read/Write	Unsigned16	6	2	instance ID of remote FB
5	RemoteObjectID	Read/Write	Unsigned16	8	2	ID of remote element object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
6	ServiceOperation	Read/Write	Unsigned8	10	1	Type 14 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 Type 14 device, then this property can be ignored; if the Type 14 service use the broadcast or multicast method, then this property should be broadcast or multicast group address
9	SendTimeOffset	Read/Write	PrecisionTime Difference	16	8	time offset when sending periodic packet from the start time of a communication macrocycle. Its data type is 4 octets of TimeDifference. The unit is nanoseconds

4.2.13 Type 14 FRT link object

The object of the Type 14 FRT Link Object class is defined as shown in Table 14.

Table 14 – Definition of Type 14 FRT 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 Type 14 Link Object in the MOB
2	Local Object ID	Read/Write	Unsigned16	2	2	the index of local variable object
3	RemoteObjectID	Read/Write	Unsigned16	4	2	ID of remote element object
4	ServiceOperation	Read/Write	Unsigned8	6	1	Type 14 service ID used by Link Object
5	ServiceRole	Read/Write	Unsigned8	7	1	role of local object in the communication process
6	RemoteMACAddress	Read/Write	Unsigned32	8	4	MAC address of remote device; if local and destination FB instance objects are in the same Type 14 device, then this property can be ignored; if the Type 14 service uses the broadcast or multicast method, then this property should be broadcast or multicast group address
7	SendTimeOffset	Read/Write	PrecisionTime Difference	12	8	time offset when sending periodic packet from the start time of a communication macrocycle. Its data type is 4 octets of TimeDifference. The unit is nanoseconds
8	ValidBitOffset	Read/Write	Unsigned16	20	4	the bit offset when the relevant message should be sent or received from the start time of field of DATA in FRTVariableDistribute Service
9	ValidBitNumber	Read/Write	Unsigned16	24	4	the bit number when the relevant message should be sent or received from the start time of field of DATA in FRTVariableDistribute Service

4.2.14 Domain application information object

The object of the Domain Application Information class is defined as shown in Table 15.

Table 15 – Definition of domain application information object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length	Description
1	Object ID	Read Only	Unsigned16	0	2	the index of the domain application information object in the MOB
2	Domain Object ID	Read Only	Unsigned16	2	2	the index of the domain object corresponding to the domain application information object
3	ConfigurationStatus	Read Only	Boolen	4	1	the configuration status of domain object, Boolean type, if its value is TRUE, then it shows that the domain object is not configured
4	Reserved	Read Only	OctetString	5	3	reserved
5	Domain Name	Read Only	Unsigned16	8	32	the name of the domain object, its length is 32 octets, the unused part is padded with BLANK (0x20)

4.3 Definition of objects used in Type 14 application access entity

Subclause 4.3 defines the encodings of objects in Type 14 application access entity.

4.3.1 Domain object

The object of the Domain class is defined as shown in Table 16.

Table 16 – Definition of domain object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	ObjectID	Read Only	Unsigned16	2	the index of Domain Object
2	Domain Name	Read/Write	VisibleString	32	the name of Domain Object
3	Max Octets	Read Only	Unsigned16	2	the maximum octets in the domain
4	Password	Read/Write	Unsigned16	2	the password used to access the Domain Object
5	AccessGroups	Read/Write	Unsigned8	1	the access group of Domain Object
6	AccessRights	Read/Write	Unsigned8	1	the access right of Domain Object
7	Local Address	Read Only	Unsigned32	4	the pointer pointed to the specific Domain Object. If not used, its value should be set to 0xFFFF FFFF
8	Domain State	Read Only	Unsigned8	1	the status of domain object, it can be the following value: 0: EXISTENT 1: DOWNLOADING 2: UPLOADING

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
					3: READY 4: IN-USE
9	Last State	Read Only	Unsigned8	1	the status of domain object before upload/download, the meaning of its value if shown as follows: 0: EXISTENT 1: DOWNLOADING 2: UPLOADING 3: READY 4: IN-USE
10	Used Application Counter	Read Only	Unsigned16	2	the number of programs using the domain now, if the counter value is bigger than 0, it shows that this domain is being used, so it cannot be overwritten by the download service, its data type is unsigned16

4.3.2 Simple variable object

The definition of Simple Variable Object is shown in Table 17.

Table 17 – Definition of simple variable object

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

4.3.3 Event object

The definition of Event Object is shown in Table 18.

Table 18 – Definition of event object

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
1	ObjectID	Read Only	Unsigned16	2	the ID of the Event Object
2	Length	Read Only	Unsigned16	2	the octet length of the Event Object
3	Password	Read/Write	Unsigned16	2	the password used to access the Domain Object
4	AccessGroups	Read/Write	Unsigned8	1	the access group of the Domain Object
5	AccessRights	Read/Write	Unsigned8	1	the access right of the Domain Object
6	Local Address	Read Only	Unsigned32	4	the pointer pointed to the specific Event

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
					Object, it is used to internally address the Variable Object. If not used , its value should be set to 0xFFFF FFFF
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 cannot be sent out

4.3.4 Type 14 socket mapping object

The definition of Type 14 Socket Mapping Object is shown in Table 19.

Table 19 – Definition of Type 14 socket mapping object

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

4.3.5 Type 14 socket timer object

The definition of the Type 14 Socket Timer Object is shown in Table 20.

Table 20 – Definition of Type 14 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

No.	Parameter name	Read/write property	Data type	Octet offset	Octet length
4	ActiveMessageTime	Read Only	Unsigned32	4	the response time of the active message, this parameter shows the maximum response time of the active message, if it does not need the response, it shall be set to zero. The unit is millisecond. The unit is milliseconds for RT applications, and microseconds for FRT applications

4.3.6 ErrorType object

The definition of the ErrorType Object is shown in Table 21.

Table 21 – Definition of ErrorType object

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

5 Transfer syntax

5.1 Encoding of basic data types

5.1.1 Boolean

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

If the value is TRUE, the value of each bit is shown in Table 22 (Bit 7 is MSB, Bit 0 is LSB).

Table 22 – Encoding of Boolean value TRUE

Octet	Bit							
	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1

If the value is FALSE, the value of each bit is shown in Table 23.

Table 23 – Encoding of Boolean value FALSE

Octet	Bit							
	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0

5.1.2 Unsigned8

The Unsigned8 type is encoded in one octet, the range is from 0 to 255, the weight of each bit is shown in Table 24.

Table 24 – Encoding of Unsigned8 data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.3 Unsigned16

The Unsigned16 type is encoded in two octets, the range is from 0 to 216-1, the weight of each bit is shown in Table 25.

Table 25 – Encoding of Unsigned16 data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.4 Unsigned32

The Unsigned32 type is encoded in four octets, the range is from 0 to 232-1, the weight of each bit is shown in Table 26.

Table 26 – Encoding of Unsigned32 data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.5 Unsigned64

The Unsigned64 type is encoded in eight octets, the range is from 0 to 264-1, the weight of each bit is shown in Table 27.

Table 27 – Encoding of Unsigned64 data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}

Octet	Bit							
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.6 Int8

The Int8 type is encoded in one octet, the range is from -128 to 127. If the sign bit (SN) is 1, the data is negative, otherwise, the data is positive or zero when bit SN is 0. The weight of each bit is shown in Table 28.

Table 28 – Encoding of Int8 data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.7 Int16

The Int16 type is encoded in two octets, the range is from -2^{15} to $2^{15}-1$. 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 Table 29.

Table 29 – Encoding of Int16 data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.8 Int32

The Int32 type is encoded in four octets, the range is from -2^{31} to $2^{31}-1$. 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 Table 30.

Table 30 – Encoding of Int32 data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.9 Int64

The Int64 type is encoded in eight octets, the range is from -2^{63} to $2^{63}-1$. 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 Table 31.

Table 31 – Encoding of Int64 data type

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

5.1.10 Real

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

Table 32 – Encoding of Real type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
2	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

5.1.11 VisibleString

The VisibleString type is encoded in visible string, the length is variable. The definition shall be according to ISO/IEC 646 and ISO/IEC 2375. The encoding is shown in Table 33.

Table 33 – Encoding of VisibleString data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	the first character							
2	the second character							
……	and so on……							
……	so on……							
N								

5.1.12 OctetString

The OctetString type is encoded in one or several octets which are aligned from 1 to n according to the sequence number. The encoding is shown in Table 34.

Table 34 – Encoding of OctetString data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	Binary data							
2	Binary data							
……	……							
……	……							
N								

5.1.13 BitString

The BitString type is encoded in a group of Octets, the length is variable from 1 to n, n is an arbitrary natural number. The encoding is shown in Table 35.

Table 35 – Encoding of BitString data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
……	so on……							
……	so on……							
n								

5.1.14 TimeOfDay

The TimeOfDay type consists of a date and a time, it is encoded in total 6 octets. The date field is encoded as Unsigned16 data (octet 1 and octet 2), it is stated in days relatively to the first of January 2000. On the first of January 2000, the date starts with the value zero. The time is stated in milliseconds since midnight, at midnight the counting starts with the value zero. The time field is encoded as an Unsigned32 data (from octet 3 to octet 6). The encoding is shown in Table 36.

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

Table 36 – Encoding of TimeOfDay data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
3	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
4	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.15 BinaryDate

The type BinaryDate consists of a calendar date and a time, it is encoded in 8 octets shown in Table 37.

The year field is encoded as an Unsigned16 data (octet 1 and octet 2), if its value is 2004, it represents the year of 2004.

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

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

The hour field is encoded as an Unsigned8 data (octet 5), the range from 0 to 23, its value represents 1 h of 24 h in a day.

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

The millisecond field is encoded as an Unsigned16 data (octet 7 and octet 8), the range is from 0 to 59999, its value represents 1 ms of 60 000 ms in a minute.

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

Table 37 – Encoding of BinaryDate data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
5	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.16 PrecisionTimeDifference

The type PrecisionTimeDifference consists of second and nanosecond, it is encoded in 8 octets ; its value states the time difference.

The Second field is encoded as an Unsigned32 data (form octet 1 to octet 4), its value represents the seconds difference.

The Nanosecond field is stated in macroseconds and encoded as an Int32 data (from octet 5 to octet 8), its value represents the macroseconds difference. The sign of Nanosecond belongs to the data type. The encoding is shown in Table 38.

```
{
    Unsigned32  Second;          //second difference
    Int32  Nanosecond;           // Nanosecond with sign
}
```

Table 38 – Encoding of PrecisionTimeDifference data type

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
5	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.17 Encoding of SEQUENCE

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

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

5.1.18 Encoding of CHOICE

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

5.2 Encoding of Type 14 APDU header

The encoding of the Type 14 Application layer Service Message packet header is shown in Table 39.

Table 39 – Encoding of Type 14 application layer service message header

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

No.	Parameter Name	Data type	Octet offset	Octet length	Description
					the whole message
4	MessageID	Unsigned16	6	2	this parameter describes the ID of the message

5.3 Encoding of FAL management entity service parameters

5.3.1 EM_DetectingDevice service

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

Table 40 – Encoding of EM_DetectingDevice request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Query Type	Unsigned8	0	1	signifies the query request type: 0: Query according to PD_Tag. The following parameter is PD_Tag 1: Query according to FB Tag. The following parameter is FB Tag 2: Query according to ElementID. The following parameter is FB Tag and ElementID
2	Reserved	Octetstring	1	3	reserved
3	PD_Tag	VisibleString	4	32	the physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)
4	FB Tag	VisibleString	36	32	function block instance tag which is used to query the information of Type 14 device which include the FB instance
5	Element ID	Unsigned16	68	2	Element ID in FB which must be used with FB Tag together, because ElementID is unique only in one FB instance

5.3.2 EM_OnlineReply service

The EM_OnlineReply request parameters are coded as shown in Table 41.

Table 41 – Encoding of EM_OnlineReply request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Query Type	Unsigned8	0	1	signifies the Query Type: 0: Query according to PD_Tag 1: Query according to FB Tag 2: Query according to ElementID
2	Duplicate Tag Detected	Boolean	1	1	this property describes if the device's PD_Tag collides with the other devices' PD_Tag (i.e. repeated PD_Tag). TRUE= PD_Tags Collide
3	Reserved	Octetstring	2	2	reserved
4	Queried Object IP Address	Unsigned32	4	4	IP address of the queried Type 14 physical device (i.e. the IP address of the local device)
5	Queried Object Device ID	VisibleString	8	32	the queried Type 14 physical device ID (e.g. local device ID), its length is 32 octets, and the unused part is padded with BLANK (0x20)
6	Queried	VisibleString	40	32	the queried Type 14 physical device tag (e.g.

No.	Parameter name	Data type	Octet offset	Octet length	Description
	Object PD_Tag				local device tag). Its length is 32 octets, and the unused part is padded with BLANK (0x20)

5.3.3 EM_GetDeviceAttribute service

5.3.3.1 Request primitive

The EM_GetDeviceAttribute request parameters are coded as shown in Table 42.

Table 42 – Encoding of EM_GetDeviceAttribute request parameters

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

5.3.3.2 Positive response primitive

The EM_GetDeviceAttribute positive response parameters are coded as shown in Table 43.

Table 43 – Encoding of EM_GetDeviceAttribute positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device ID	VisibleString	0	32	Local device ID
2	PD_Tag	VisibleString	32	32	the physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)
3	Status	Unsigned8	64	1	the status of the Type 14 device: 0: no address 1: unconfigured 2: configured, operational
4	Device Type	Unsigned8	65	1	local device type
5	Annunciation Interval	Unsigned16	66	2	the interval of the annunciation message sent out by the device
6	Annunciation Version Number	Unsigned16	68	2	the version number of the annunciation message
7	Duplicate Tag Detected	Boolean	70	1	this property indicates whether PD_Tag of the device is in collision with the other or not (i.e. duplicated PD_Tag). TRUE= PD_Tag in collision
8	Redundancy Number	Unsigned8	71	1	the redundancy number of local device, if the device is active, then this value is zero, and has no following parameters
9	Device Redundancy State	Unsigned8	72	1	the redundancy status the local device is in: 0: active status 1: back-up status If the redundancy number is 0, then the response primitive does not contain this parameter

No.	Parameter name	Data type	Octet offset	Octet length	Description
10	Max Redundancy Number	Unsigned8	73	1	the maximum redundancy number of the device, if the redundancy number is 0, then the response primitive does not contain this parameter
11	Reserved	Octetstring	74	2	reserved
12	Active IP Address	Unsigned32	76	4	IP address of the active device (if no redundancy, then its local IP address); if the redundancy number is 0, then the response primitive does not contain this parameter

5.3.3.3 Negative response primitive

The EM_GetDeviceAttribute negative response parameters are coded as shown in Table 44.

Table 44 – Encoding of EM_GetDeviceAttribute negative response parameters

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

5.3.4 EM_ActiveNotification service

The EM_ActiveNotification request parameters are coded as shown in Table 45.

Table 45 – Encoding of EM_ActiveNotification request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device ID	VisibleString	0	32	ID of local device
2	PD_Tag	VisibleString	32	32	the local physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)
3	Status	Unsigned8	64	1	the status of the Type 14 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 an active device, then this value is 0, otherwise the following parameter is invalid
7	Device Redundancy State	Unsigned8	69	1	the redundancy status of the local device: 0: active state 1: backup state If the redundancy number is 0, then the response primitive does not contain this parameter
8	LAN Redundancy Port	Unsigned16	70	2	the LAN redundancy message processing port of the device which requests the service
9	Duplicate Tag Detected	Boolean	72	1	this property describes if the device's PD_Tag collides with the other devices' PD_Tag (i.e. duplicated PD_Tag). TRUE= PD_Tags Collide
10	Reserved	Octetstring	73	2	reserved
11	Max. Redundancy Number	Unsigned8	75	1	the maximum redundancy number of the device
12	Active IP Address	Unsigned32	76	4	IP address of the active device (if no redundancy, then its local IP address); if the redundancy number is 0, then the response primitive does not contain this parameter

5.3.5 EM_ConfiguringDevice service

5.3.5.1 Request Primitive

The EM_ConfiguringDevice request parameters are coded as shown in Table 46.

Table 46 – Encoding of EM_ConfiguringDevice request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationIPAddress	Unsigned32	0	4	Destination IP Address
2	Device ID	VisibleString	4	32	ID of local device
3	PD_Tag	VisibleString	36	32	the local physical device tag, its length is 32 octets, and the unused part is padded with BLANK (0x20)
4	Annunciation Interval	Unsigned16	68	2	the interval of the annunciation message sent out by the device
5	Duplicate Tag Detected	Boolean	70	1	this property describes if the device's PD_Tag collides with the other devices' PD_Tag (i.e. duplicated PD_Tag). TRUE= PD_Tags Collide
6	Device Redundancy Number	Unsigned8	71	1	the redundancy number of the local device, if its active device, then this value is 0, and does not have the following parameters
7	LAN Redundancy Port	Unsigned16	72	2	the LAN redundancy message processing port of the device which requests the service
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	76	4	IP address of the active device (if no redundancy, then its local IP address); if the redundancy number is 0, then the response primitive does not contain this parameter

5.3.5.2 Positive response primitive

The EM_ConfiguringDevice positive response parameters are coded as shown in Table 47.

Table 47 – Encoding of EM_ConfiguringDevice positive response parameters

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

5.3.5.3 Negative response primitive

The EM_ConfiguringDevice negative response parameters are coded as shown in Table 48.

Table 48 – Encoding of EM_ConfiguringDevice negative response parameters

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

5.3.6 EM_SetDefaultValue service

5.3.6.1 Request primitive

The EM_SetDefaultValue request parameters are coded as shown in Table 49.

Table 49 – Encoding of EM_SetDefaultValue request parameters

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

5.3.6.2 Positive response primitive

The EM_SetDefaultValue positive response parameters are coded as shown in Table 50.

Table 50 – Encoding of EM_SetDefaultValue positive response parameters

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

5.3.6.3 Negative response primitive

The EM_SetDefaultValue negative response parameters are coded as shown in Table 51.

Table 51 – Encoding of clear device attribute service refuse packet

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

5.4 Encoding of AAE Services

5.4.1 DomainDownload Service

5.4.1.1 Request primitive

The DomainDownload request parameters are coded as shown in Table 52.

Table 52 – 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 from 0 to 512
8	Load Data	OctetString	12	N	domain download data

5.4.1.2 Positive response primitive

The DomainDownload positive response parameters are coded as shown in Table 53.

Table 53 – Encoding of domain download service response packet

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

5.4.1.3 Negative response primitive

DomainDownload negative response parameters are coded as shown in Table 54.

Table 54 – Encoding of DomainDownload negative response parameters

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

5.4.2 Domain Upload Service

5.4.2.1 Request primitive

The DomainUpload request parameters are coded as shown in Table 55.

Table 55 – Encoding of DomainUpload request parameters

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

5.4.2.2 Positive response primitive

The DomainUpload positive response parameters are coded as shown in Table 56.

Table 56 – 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 range is from 0 to 512
3	MoreFollows	Boolean	4	1	flag used to indicate whether there are more downloads data
4	Reserved	Octetstring	5	3	reserved
5	Load Data	Octetstring	8	N	domain upload data

5.4.2.3 Negative response primitive

The DomainUpload negative response parameters are coded as shown in Table 57.

Table 57 – Encoding of DomainUpload negative response parameters

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

5.4.3 EventRoport Service

The EventRoport request parameters are coded as shown in Table 58.

Table 58 – Encoding of EventRoport request parameters

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

5.4.4 EventRoportAcknowledge Service

5.4.4.1 Request primitive

The EventRoportAcknowledge request parameters are coded as shown in Table 59.

Table 59 – Encoding of EventRoportAcknowledge request parameters

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

5.4.4.2 Positive response primitive

The EventRoportAcknowledge positive response parameters are coded as shown in Table 60.

Table 60 – Encoding of EventRoportAcknowledge positive response parameters

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

5.4.4.3 Negative response primitive

The EventRoportAcknowledge negative response parameters are coded as shown in Table 61.

Table 61 – Encoding of EventRoportAcknowledge negative response parameters

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

5.4.5 ReportConditionChanging service

5.4.5.1 Request primitive

The ReportConditionChanging request parameters are coded as shown in Table 62.

Table 62 – Encoding of ReportConditionChanging request parameters

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

5.4.5.2 Positive response primitive

The ReportConditionChanging positive response parameters are coded as shown in Table 63.

Table 63 – Encoding of ReportConditionChanging positive response parameters

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

5.4.5.3 Negative response primitive

The ReportConditionChanging negative response parameters are coded as shown in Table 64.

Table 64 – Encoding of ReportConditionChanging negative response parameters

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

5.4.6 Read service

5.4.6.1 Request primitive

The Read request parameters are coded as shown in Table 65.

Table 65 – Encoding of Read request parameters

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

5.4.6.2 Positive response primitive

The Read positive response parameters are coded as shown in Table 66.

Table 66 – Encoding of Read positive response parameters

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

5.4.6.3 Negative response primitive

The Read negative response parameters are coded as shown in Table 67.

Table 67 – Encoding of Read negative response parameters

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

5.4.7 Write Service

5.4.7.1 Request primitive

The Write request parameters are coded as shown in Table 68.

Table 68 – Encoding of Write request parameters

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

5.4.7.2 Positive response primitive

The Write positive response parameters are coded as shown in Table 69.

Table 69 – Encoding of Write positive response parameters

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

5.4.7.3 Negative response primitive

The Write negative response parameters are coded as shown in Table 70.

Table 70 – Encoding of Write negative response parameters

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

5.4.8 VariableDistribute Service

The VariableDistribute request parameters are coded as shown in Table 71.

Table 71 – Encoding of VariableDistribute request parameters

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

5.4.9 FRTRead service

5.4.9.1 Request primitive

The FRTRead request parameters are coded as shown in Table 72.

Table 72 – Encoding of FRTRead request parameters

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

5.4.9.2 Positive response primitive

The FRTRead positive response parameters are coded as shown in Table 73.

Table 73 – Encoding of FRTRead positive response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationObjectID	Unsigned16	0	2	object ID of destination device
2	Reserved	Octetstring	2	2	reserved
3	Data	Octetstring	4	N	returned data

5.4.9.3 Negative response primitive

The FRTRead negative response parameters are coded as shown in Table 74.

Table 74 – Encoding of FRTRead negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationObjectID	Unsigned16	0	2	object ID of destination device
2	Reserved	Octetstring	2	2	reserved
3	Error Type	ErrorType	4	N	see Error Type

5.4.10 FRTWrite Service

5.4.10.1 Request primitive

The FRTWrite request parameters are coded as shown in Table 75.

Table 75 – Encoding of FRTWrite request parameters

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

5.4.10.2 Positive response primitive

The FRTWrite positive response parameters are coded as shown in Table 76.

Table 76 – Encoding of FRTWrite positive response parameters

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

5.4.10.3 Negative response primitive

The FRTWrite negative response parameters are coded as shown in Table 77.

Table 77 – Encoding of FRTWrite negative response parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	DestinationOjbectID	Unsigned16	0	2	object ID of destination device
2	Reserved	Octetstring	2	2	reserved
3	Error Type	ErrorType	4	N	see Error Type

5.4.11 FRTVariableDistribute Service

The FRTVariableDistribute request parameters are coded as shown in Table 78.

Table 78 – Encoding of FRTVariableDistribute request parameters

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

5.4.12 BlockTransmissionOpen service

5.4.12.1 Request primitive

The BlockTransmissionOpen request parameters are coded as shown in Table 79.

Table 79 – Encoding of BlockTransmissionOpen request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	SourceAppID	Unsigned16	0	2	application ID of source device
2	DestinationAppID	Unsigned16	2	2	application ID of destination device
3	DestinationObjectID	Unsigned16	4	2	object ID of destination device
4	BlockType	Unsigned16	6	2	the type of block data 0: video; 1: audio; Other: Reserved
5	BlockConfigInfo	OctetString	8	N	the configuration information of the block transmission

5.4.12.2 Positive response primitive

The BlockTransmissionOpen positive response parameters are coded as shown in Table 80.

Table 80 – Encoding of BlockTransmissionOpen positive response parameters

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

5.4.12.3 Negative response primitive

The BlockTransmissionOpen negative response parameters are coded as shown in Table 81.

Table 81 – Encoding of BlockTransmissionOpen negative response parameters

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

5.4.13 BlockTransmissionClose service

5.4.13.1 Request primitive

The BlockTransmissionClose request parameters are coded as shown in Table 82.

Table 82 – Encoding of BlockTransmissionClose request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	SourceAppID	Unsigned16	0	2	application ID of source device
2	DestinationAppID	Unsigned16	2	2	application ID of destination device
3	DestinationObjectID	Unsigned16	4	2	object ID of destination device
4	BlockType	Unsigned16	6	2	the type of block data 0: video; 1: audio; Other: Reserved

5.4.13.2 Positive response primitive

The BlockTransmissionClose positive response parameters are coded as shown in Table 83.

Table 83 – Encoding of BlockTransmissionClose positive response parameters

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

5.4.13.3 Negative response primitive

The BlockTransmissionClose negative response parameters are coded as shown in Table 84.

Table 84 – Encoding of BlockTransmissionClose negative response parameters

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

5.4.14 BlockTransmit Service

The BlockTransmit request parameters are coded as shown in Table 85.

Table 85 – Encoding of BlockTransmit request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	SourceAppID	Unsigned16	0	2	application ID of source device
2	DestinationAppID	Unsigned16	2	2	application ID of destination device
3	DestinationObjectID	Unsigned16	4	2	object ID of destination

No.	Parameter name	Data type	Octet offset	Octet length	Description
					device
4	DataLength	Unsigned16	6	2	Data length
5	BlockType	Unsigned16	8	2	The type of block data
6	SequenceNumber	Unsigned16	10	2	Sequence number
7	TimeStamp	PrecisionTimeDifference	12	8	Time stamp
8	SendCount	Unsigned16	20	2	Total number of sent packets
9	BlockData	Octetstring	22	N	The block data

5.4.15 BlockTransmissionHeartbeat Service

The BlockTransmissionHeartbeat request parameters are coded as shown in Table 86.

Table 86 – Encoding of BlockTransmissionHeartbeat request parameters

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	SourceAppID	Unsigned16	0	2	application ID of source device
2	DestinationAppID	Unsigned16	2	2	application ID of destination device
3	DestinationObjectID	Unsigned16	4	2	object ID of destination device
4	ReceptionCount	Unsigned16	6	2	the total number of the received packets
5	CumulativeLost	Unsigned16	8	2	the total number of block data packets that have been lost
6	Jitter	PrecisionTimeDifference	10	8	the transmission jitter calculated from the time stamp

6 Structure of FAL protocol state machines

Interface to FAL services and protocol machines are specified in Clause 6. Conventions used for the descriptions are given in the generic part of this standard.

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

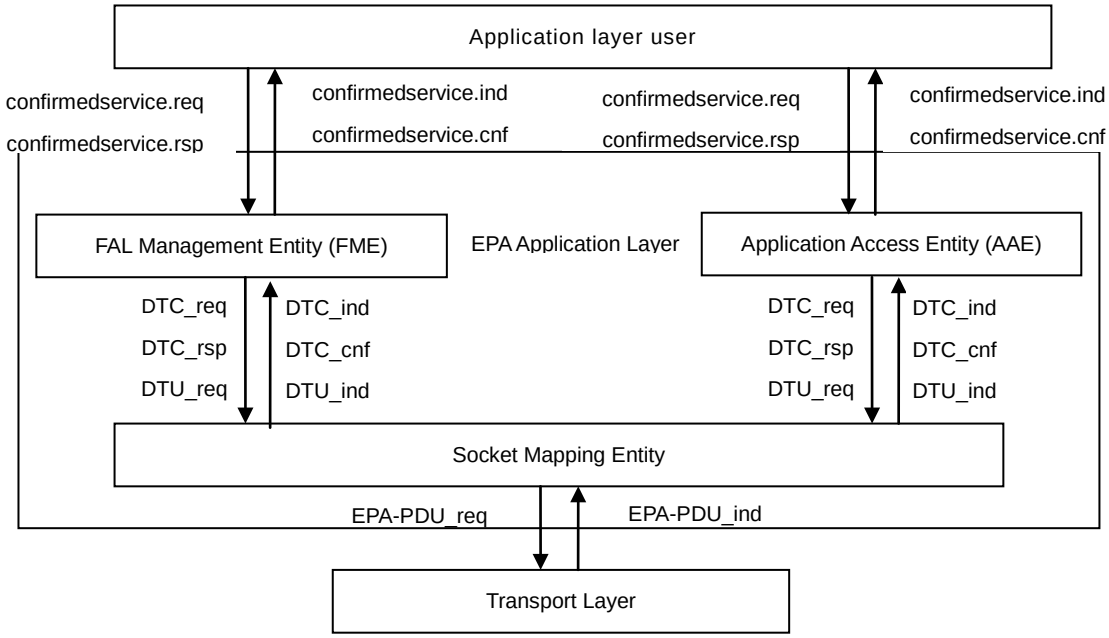


Figure 2 – Exchanged primitives of protocol state machine

7 AP-Context state machine

7.1 Primitives exchanged between ALU and ALE

Table 87 and Table 88 show the primitives exchanged between the Application Layer User (ALU) and Application Layer Entity (ALE).

Table 87 – Primitives delivered by ALU to ALE

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

Table 88 – Primitives delivered by ALE to ALU

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

7.2 Protocol state machine descriptions

The AP Context Entity of Type 14 is always active. Its state is described in Table 89.

Table 89 – ACE state descriptions

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

7.3 State transitions

The state transitions of ACE are defined in Figure 3, Table 90 and Table 91.

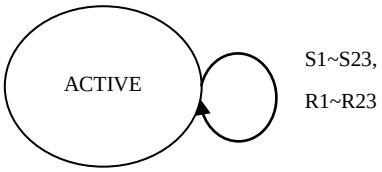


Figure 3 – ACE protocol state machine

Table 90 – ACE state transitions (sender)

#	Current state	Event or condition => action	Next state

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

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

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

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

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

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

#	Current state	Event or condition => action	Next state
		}	

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

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

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

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

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

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

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

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

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

#	Current state	Event or condition => action	Next state
		}	

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

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

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

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

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

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

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

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

Table 91 – ACE state transitions (receiver)

#	Current state	Event or condition => action	Next state

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

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

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

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

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

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

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

R8	ACTIVE	Confirmedservice.ind APServiceType(data) = “EM_SetDeviceAttribute” =>	ACTIVE

#	Current state	Event or condition => action	Next state

		EM_SetDeviceAttribute.ind {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}	

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

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

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

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

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

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

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

#	Current state	Event or condition => action	Next state

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

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

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

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

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

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

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

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

#	Current state	Event or condition => action	Next state
		Destination_ip : = remote_ip_address, }	

7.4 Function descriptions

Table 92 describes the functions used by ACE state transitions.

Table 92 – APServiceType() descriptions

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

8 FAL management state machines

8.1 Primitives

8.1.1 Primitives exchanged between FME and application layer user

Table 93 and Table 94 show the primitives exchanged between FME and application layer user.

Table 93 – Primitives delivered by application layer user to FME

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

Table 94 – Primitives delivered by FME to application layer user

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

8.1.2 Primitive parameters of FME and user of application layer

Table 95 shows the primitives parameters of FME and the user of application layer.

Table 95 – Primitive parameters exchanged between FME and application layer user

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

8.1.3 Primitives exchanged between FME and ESME

Table 96 and Table 97 show the primitives exchanged between FME and ESME.

Table 96 – Primitives delivered by FME to ESME

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

Table 97 – Primitives delivered by ESME to FME

Primitive name	Source	Associated parameters	Functions
DTC_ind	Type 14 Socket Mapping Entity	remote_ip_address, Data	This primitive is used to send a confirmed service indication primitive to FME by ESME
DTC_cnf	Type 14 Socket Mapping Entity	remote_ip_address, Data	This primitive is used to send a confirmed service acknowledge primitive to FME by ESME
DTU_ind	Type 14 Socket Mapping Entity	remote_ip_address, Data	This primitive is used to send a unconfirmed service indication primitive to FME by ESME

8.1.4 Primitive parameters exchanged between FME and ESME

Table 98 shows the primitives parameters exchanged between FME and ESME.

Table 98 – Primitives parameters exchanged between FME and ESME

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

8.2 Protocol state machine descriptions

The states of Type 14 devices can be one of *No address*, *Unconfigured* or *Configured*. The protocol state machine is shown in Figure 4.

8.2.1 No address

In this state, the Type 14 device waits for IP address assignment. The IP address can be appointed statically by user or assigned by DHCP server. After the device gets its IP address through DHCP, it will change its next state into *Unconfigured* or *Configured* depending on its state when it was powered off before. That is, if the state when the device powered off before is *Configured*, then the next state is *Configured*, vice versa.

8.2.2 Unconfigured

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

8.2.3 Configured

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

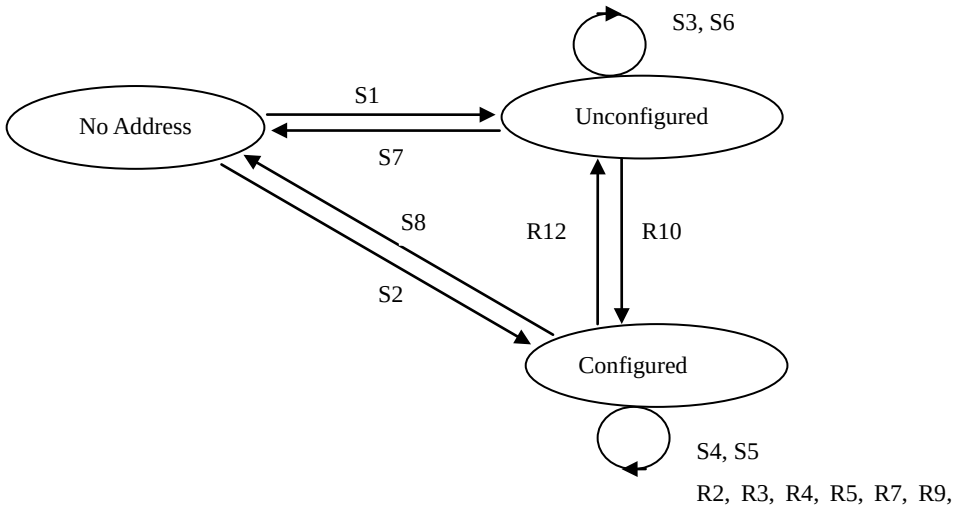


Figure 4 – FME protocol state machine

8.3 State transitions

The state transitions of FME are defined in Figure 4 and Table 99.

Table 99 – State transitions of Type 14 FME

#	Current state	Event or condition => action	Next state
S1	No Address	RcvNewIpAddress (address) = TRUE && Attribute_Set ()= FALSE => RestoreDefaults () NewAddress(address) EM_ActiveNotification.req {} Restart_Type 14RepeatTimer()	Unconfigured
S2	No Address	RcvNewIpAddress (address) = TRUE && Attribute_Set()=TRUE => Clear_DuplicatePdTagFlag() NewAddress (address) EM_ActiveNotification.req {} Restart_Type 14RepeatTimer () EM_DetectingDevice.req {}	Configured
S3	Unconfigured	RepeatTimerExpires () =>	Unconfigured

#	Current state	Event or condition => action	Next state
		EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	
S4	Configured	EM_UnconfirmedService.req {} || EM_ConfirmedService.req {} || EM_ConfirmedService.rsp {} => Send_EM_ReqRspMessage (em_svc)	Configured
S5	Configured	EM_ConfirmedService.err {} => Send_EM_CommonErrorRsp ()	Configured
S6	Unconfigured	SntpSyncLost () => EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	Unconfigured
S7	Unconfigured	IPAddressCollision () = TRUE = > (no actions taken)	No Address
S8	Configured	IPAddressCollision () = TRUE = > (no actions taken)	No Address
R1	Unconfigured	RecvMsg () =”Any Confirmed Type 14_FME Rsp Message” || RecvMsg ()=”Any Confirmed Type 14_FME Error Message” => EM_ConfirmedService.cnf{}	Unconfigured
R2	Configured	RecvMsg()=” EM_DetectingDevice” && QueryMatch (em_svc) = TRUE => EM_OnlineReply.req {}	Configured
R3	Configured	RecvMsg()=” EM_OnlineReply” && _MessageIdMatch(em_svc) = TRUE && DeviceId_Match (em_svc) = FALSE => Type 14Set_DuplicatePdTagFlag () EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	Configured
R4	Configured	RecvMsg()=” EM_OnlineReply” && _MessageIDMatch(em_svc) = TRUE && DeviceId_Match (em_svc) = TRUE => //Do nothing-the response is from this device	Configured
R5	Configured	RecvMsg() = ” EM_OnlineReply” => EM_OnlineReply.ind {}	Configured
R6	Unconfigured	RecvMsg() = ” EM_GetDeviceAttributeReq” => EM_ConfrimedService.err {}	Unconfigured
R7	Configured	RecvMsg()=” EM_GetDeviceAttributeReq” => EM_GetDeviceAttribute.rsp {}	Configured

#	Current state	Event or condition => action	Next state
R8	Unconfigured	RecvMsg()=" EM_ConfiguringDeviceReq" || RecvMsg()=" EM_SetDefaultValueReq" && DeviceId_Match(em_svc)= FALSE => EM_ConfirmedService.err { }	Unconfigured
R9	Configured	RecvMsg()="EM_ConfiguringDeviceReq" || RecvMsg()=" EM_SetDefaultValueReq" && PdTag_Match(em_svc)= TRUE => EM_ConfirmedService.rsp {}	Configured
R10	Unconfigured	RecvMsg()=" EM_ConfiguringDeviceReq" => Set_Attribute_Data(em_svc) Clear_DuplicatePdTagFlag () EM_ConfiguringDevice.rsp {} EM_ActiveNotification.req {} Restart_Type 14RepeatTimer () EM_DetectingDevice.req {}	Configured
R11	Configured	RecvMsg=" EM_GetDeviceAttributeReq" => EM_GetDeviceAttribute.rsp {}	Configured
R12	Configured	RecvMsg()="EM_SetDefaultValueReq" => ResoreDefaults () EM_SetDefaultValue.rsp {} EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	Unconfigured

8.4 Function descriptions

The functions used in FME state transitions are listed in Table 100 through Table 117.

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

8.4.1 RcvNewIpAddress()

The RcvNewIpAddress() is illustrated in Table 100.

Table 100 – RcvNewIpAddress() descriptions

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

8.4.2 Attribute_Set()

The Attribute_Set() is illustrated in Table 101.

Table 101 – Attribute_Set() descriptions

Name	Attribute_Set	Using	FME
Input		Output	
None		TRUE or FALSE	
Function			
Returns TRUE if the attribute of the Type 14 device has been set by the EM_ConfiguringDevice service and is currently still valid. Otherwise, returns FALSE			

8.4.3 RestoreDefaults()

The RestoreDefaults() is illustrated in Table 102.

Table 102 – RestoreDefaults() descriptions

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

8.4.4 NewAddress()

The NewAddress() is illustrated in Table 103.

Table 103 – NewAddress() descriptions

Name	NewAddress	Using	FME
Input		Output	
Address		TRUE or FALSE	
Function			
Returns TRUE if the IP address is updated correctly when Type 14 device received a new IP address. Otherwise, returns FALSE			

8.4.5 Restart_Type 14RepeatTimer()

Restart_Type 14RepeatTimer() is illustrated in Table 104.

Table 104 – Restart_Type 14RepeatTimer() descriptions

Name	Restart_Type 14RepeatTimer	Using	FME
Input		Output	
None		None	
Function			
This function is invoked to restore and restart Type 14 RepeatTimer			

8.4.6 Clear_DuplicatePdTagFlag()

The Clear_DuplicatePdTagFlag() is illustrated in Table 105.

Table 105 – Clear_DuplicatePdTagFlag() descriptions

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

8.4.7 Type 14RepeatTimerExpire()

The Type 14RepeatTimerExpire() is illustrated in Table 106.

Table 106 – Type 14RepeatTimerExpire() descriptions

Name	Type 14RepeatTimerExpire	Using	FME
Input		Output	
None		None	
Function			
This function is invoked when the Type 14 Repeat Timer expires			

8.4.8 Send_EM_ReqRspMessage()

The Send_EM_ReqRspMessage() is illustrated in Table 107.

Table 107 – Send_EM_ReqRspMessage() descriptions

Name	Send_EM_ReqRspMessage	Using	FME
Input		Output	
Em_svc		None	
Function			
Constructs and sends request or response messages			

8.4.9 Send_EM_CommonErrorRsp()

The Send_EM_CommonErrorRsp() is illustrated in Table 108.

Table 108 – Send_EM_CommonErrorRsp() descriptions

Name	Send_EM_CommonErrorRsp	Using	FME
Input		Output	
service_type, Spec_params		None	
Function			
Constructs and sends error response information			

8.4.10 SntpSyncLost()

The SntpSyncLost() is illustrated in Table 109.

Table 109 – SntpSyncLost() descriptions

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

8.4.11 IPAddressCollision()

The IPAddressCollision() is illustrated in Table 110.

Table 110 – IPAddressCollision() descriptions

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

8.4.12 RecvMsg()

The RecvMsg() is illustrated in Table 111.

Table 111 – RecvMsg() descriptions

Name	RecvMsg	Using	FME
Input		Output	
Em_svc		Em_svc message type	
Function			
This function is invoked when a message is received. It decodes the em_svc and returns the type of the Type 14 management service			

8.4.13 QueryMatch()

The QueryMatch() is illustrated in Table 112.

Table 112 – QueryMatch() descriptions

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

8.4.14 MessageIDMatch()

The MessageIDMatch() is illustrated in Table 113.

Table 113 – MessageIDMatch() descriptions

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

8.4.15 DeviceId_Match()

The DeviceId_Match() is illustrated in Table 114.

Table 114 – DevId_Match() descriptions

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

8.4.16 PdTag_Match()

The PdTag_Match() is illustrated in Table 115.

Table 115 – PdTag_Match() descriptions

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

8.4.17 Set_Attribute_Data()

The Set_Attribute_Data() is illustrated in Table 116.

Table 116 – Set_Attribute_Data() descriptions

Name	Set_Attribute_Data	Using	FME
Input		Output	
em_svc		None	
Function			
Updates the attributes of the EME in the Type 14 device			

8.4.18 Set_DuplicatePdTagFlag()

The Set_DuplicatePdTagFlag() is illustrated in Table 117.

Table 117 – Set_DuplicatePdTagFlag() descriptions

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

9 Application access entity protocol machine

9.1 Primitives

9.1.1 Primitives exchanged between AAE and application layer user

The primitives exchanged between AAE and Application Layer User (ALU) are shown in Table 118 and Table 119.

Table 118 – 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 119 – Primitives issued by AAE to ALU

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

9.1.2 Primitive parameters exchanged between AAE and ALU

Table 120 describes the parameters used with primitives exchanged between AAE and ALU.

Table 120 – Primitives parameters exchanged between AAE and ALU

Parameter name	Description
remote_ip_address	This parameter transfers 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 transfers the data that the sender will send and the receiver will receive

9.1.3 Primitives Exchanged between AAE and ESME

The primitives exchanged between AAE and ESME are shown in Table 121 and Table 122.

Table 121 – 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 122 – 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

9.1.4 Primitive parameters exchanged between AAE and ESME

Table 123 describes the parameters used with primitives exchanged between AAE and ESME.

Table 123 – Primitive parameters exchanged between AAE and ESME

Parameter name	Description
remote_ip_address	This parameter transfers 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 transfers the data that the sender will send and the receiver will receive

9.2 AAE state machine

9.2.1 AAE state description

Type 14 Application Access Entity is always active. Its state is described in Table 124.

Table 124 – AAE state descriptions

States	Descriptions
ACTIVE	The ACEs in ACTIVE state prepare to transfer service primitives to ALU and ESME, or to receive primitives from ALU and ESME

9.2.2 State transitions

The protocol state machine of AAE is illustrated in the Figure 5 and Table 125 through Table 126.

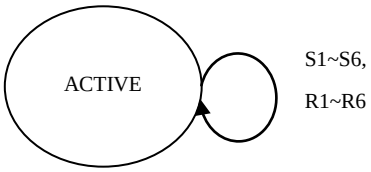


Figure 5 – AAE state transition diagrams

Table 125 – 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 {	ACTIVE

#	Current state	Event or condition => action	Next state
		user_data := Data, Destination_ip : = remote_ip_address, }	

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 126 – 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”	ACTIVE

#	Current state	Event or condition => action	Next state

		=>
		UnconfirmedService.ind {
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}	

9.2.3 Function descriptions

Table 127 describes the functions used by AAE state transitions.

Table 127 – 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 non-confirm service indication primitive			

9.3 Event ASE protocol machine

9.3.1 State description

Event ASE has two states: LOCKED and UNLOCKED described in Table 128.

Table 128 – State value of event management

States	Descriptions
LOCKED	LOCKED(Enabled=FALSE)means no event notification to transfer
UNLOCKED	UNLOCKED(Enable = TRUE)means event notification to transfer normally

9.3.2 State transitions

State transitions of Event ASE are shown in Figure 6 and Table 129.

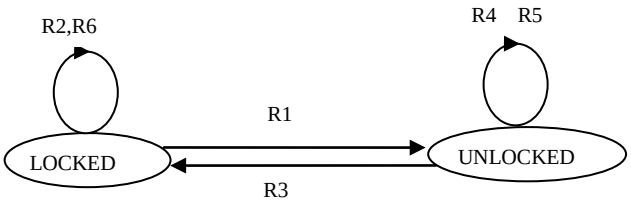


Figure 6 – Event ASE state transition diagrams

Table 129 – Event ASE state transition table

#	Current state	Event or condition => action	Next state

R1	LOCKED	ReportConditionChanging.ind
		&& Enabled = TRUE
		=>
		ReportConditionChanging.rsq(+){
		}
			UNLOCKED

R2	LOCKED	ReportConditionChanging.ind
		&&Enabled = FALSE
		=>
		ReportConditionChanging.rsq(+){
		}
			LOCKED

R3	UNLOCKED	ReportConditionChanging.ind
		&&Enabled = FALSE
		=>
		ReportConditionChanging.rsq(+){
		}
			LOCKED

R4	UNLOCKED	ReportConditionChanging.ind
		&&Enabled = TRUE
		=>
		ReportConditionChanging.rsq(+){
		}
			UNLOCKED

R5	UNLOCKED	Type 14 AL service.ind <>" ReportConditionChanging.ind" => (no actions taken)	UNLOCKED
R6	LOCKED	Type 14 AL service.ind <>" ReportConditionChanging.ind" => (no actions taken)	LOCKED

9.3.3 Function descriptions

No additional functions are used in Event ASE state transitions.

9.4 Domain ASE protocol machine

9.4.1 State descriptions

Domain ASE has five states described in Table 130.

Table 130 – 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

9.4.2 State transitions

The state transitions of Domain ASE are illustrated in Figure 7 and Table 131.

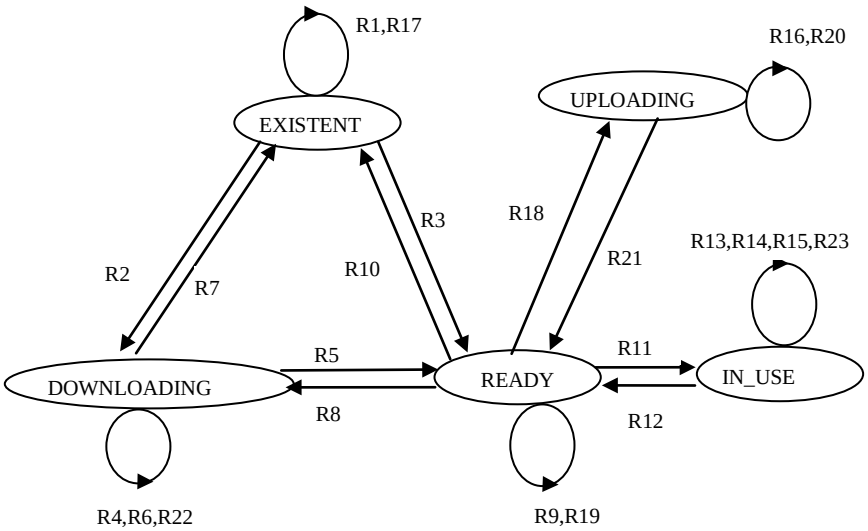


Figure 7 – Domain ASE state transition diagram

Table 131 – 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
R5	DOWNLOADING	DomainDownload.ind && Domain_DownloadSucceed() = TRUE &&MoreFollow= FALSE => DomainDownload.rsp(+){ }. Domain_WriteBuffer()	READY
R6	DOWNLOADING	DomainDownload.ind && Domain_DownloadSucceed() = FALSE => DomainDownload.err{	DOWNLOADING

	Current state	Event or condition => action	Next state
		}	

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

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	UPLOADING

	Current state	Event or condition => action	Next state

		=>
		DomainUpload.rsp(+){
		MoreFollows := TRUE
		}	

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

9.4.3 Functions description

Table 132 through Table 135 described the functions used in the Domain ASE state transitions.

9.4.3.1 Domain_DownloadSucceed() description

Table 132 describes Domain_DownloadSucceed() function.

Table 132 – Domain_DownloadSucceed() description

Name	Domain_DownloadSucceed	Used in	AAE
Input		Output	
None		TRUE or FALSE	
Function			
Judge the download state, if failed, then returns FALSE; if succeeded, then returns TRUE			

9.4.3.2 Domain_WriteBuffer() description

Table 133 describes Domain_WriteBuffer() function.

Table 133 – Domain_WriteBuffer() description

Name	Domain_WriteBuffer	Used in	AAE
Input		Output	
None		None	
Function			
Write the received data into buffer			

9.4.3.3 IncreamentInvokeDomainCounter() description

Table 134 describes IncreamentInvokeDomainCounter() function.

Table 134 – IncreamentInvokeDomainCounter() description

Name	IncreamentInvokeDomainCounter	Used in	AAE
Input		Output	
None		None	
Function			
IncreamentInvokeDomainCounter increases by 1			

9.4.3.4 DecreamentInvokeDomainCounter() description

Table 135 describes DecreamentInvokeDomainCounter() function.

Table 135 – DecreamentInvokeDomainCounter() description

Name	DecreamentInvokeDomainCounter	Used in	AAE
Input		Output	
None		None	
Function			
DecreamentInvokeDomainCounter decreases by 1			

9.5 Block ASE protocol machine

9.5.1 State description

Block ASE has two states: IDLE and TRANSMISSION described in Table 136.

Table 136 – State value of Block transmission

States	Descriptions
IDLE	The device has no block transmitting mission currently
TRANSMITTING	The device is transmitting block data currently

9.5.2 State transitions

State transitions of Block ASE are shown in Figure 8 and Table 137.

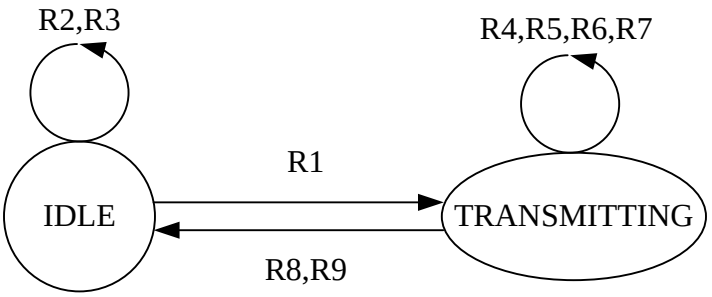


Figure 8 – Block ASE state transition diagrams

Table 137 – Block ASE state transition table

#	Current state	Event or condition => action	Next state

R1	IDLE	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = TRUE
		&& MemberNum = 0
		=>
		BlockTransmissionOpen.rsp(+){
		}
		BlockTransmit.ind{
		}
		MemberNum = 1
			TRANSMITTING

R2	IDLE	BlockTransmissionClose.ind
		=>
		BlockTransmissionClose.rsp(-){
		ErrorType:=Service Error
		}
			IDLE

R3	IDLE	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = FLASE
		=>
		BlockTransmissionOpen.err{
		}
			IDLE

R4	TRANSMITTING	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = FLASE
		=>
		BlockTransmissionOpen.err{
		}
			TRANSMITTING

R5	TRANSMITTING	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = TRUE
		&& MemberNum > 0
		=>
		BlockTransmissionOpen.rsp(+){
		}
		MemberNum = MemberNum + 1
			TRANSMITTING

R6	TRANSMITTING	BlockTransmissionClose.ind
		&& BlockTransmissionCloseSucceed() = TRUE
		&& MemberNum > 1
		=>
		BlockTransmissionClose.rsp(+){
		}
		MemberNum = MemberNum – 1
			TRANSMITTING

R7	TRANSMITTING	BlockTransmissionClose.ind && BlockTransmissionCloseSucceed() = FLASE =>	TRANSMITTING

#	Current state	Event or condition => action	Next state
		BlockTransmissionClose.err{ }	

R8	TRANSMITTING	BlockTransmissionClose.ind
		&& BlockTransmissionCloseSucceed() = TRUE
		&& MemberNum = 1
		=>
		BlockTransmissionClose.rsp(+){
		}
		MemberNum = 0
			IDLE

R9	TRANSMITTING	ReceiveBlockTransmissionHeartbeat_timeout = TRUE => ReceiveBlockTransmissionHeartbeat_timeout() MemberNum = 0	IDLE

9.5.3 Function descriptions

Table 138 through Table 140 describes the functions used by Block ASE state transitions.

Table 138 – BlockTransmissionOpenSucceed() descriptions

Name	BlockTransmissionOpenSucceed	Used in	AAE
Input		Output	
Data		TRUE or FALSE	
Function			
Judge the BlockTransmissionOpen state, if failed, then returns FALSE; if succeeded, then returns TRUE			

Table 139 – BlockTransmissionCloseSucceed() descriptions

Name	BlockTransmissionCloseSucceed	Used in	AAE
Input		Output	
Data		TRUE or FALSE	
Function			
Judge the BlockTransmissionClose state, if failed, then returns FALSE; if succeeded, then returns TRUE			

Table 140 – ReceiveBlockTransmissionHeartbeat_timeout() description

Name	ReceiveBlockTransmissionHeartbeat_timeout	Used in	AAE
Input		Output	
None		None	
Function			
End the transmission of block data if the device hasn't received BlockTransmissionHeartbeat in a certain time			

10 Application relationship state machine

10.1 Primitives

10.1.1 Primitives Exchanged between AREP and FME(or AAE)

The primitives exchanged between AREP and FME(or AAE) are shown in Table 141 and Table 142.

Table 141 – Primitives issued by FME(or AAE) to AREP

Primitive name	Source	Associated parameters	Functions
DTC_req	FME(or AAE)	remote_ip_address, data	This is a primitive used to convey a ConfirmedService request primitive from the FME(or AAE) to the AREP
DTC_rsp	FME(or AAE)	remote_ip_address, data	This is a primitive used to convey a ConfirmedService response primitive from the FME(or AAE) to the AREP
DTU_req	FME(or AAE)	remote_ip_address, data	This is a primitive used to convey a UnconfirmedService request primitive from FME(or AAE) to the AREP

Table 142 – Primitives issued by AREP to FME(or AAE)

Primitive name	Source	Associated parameters	Functions
DTC_ind	AREP	remote_ip_address, data	This is a primitive used to convey a ConfirmedService indication primitive from the AREP to the FME(or AAE)
DTC_cnf	AREP	remote_ip_address, data	This is a primitive used to convey a ConfirmedService confirmation primitive from the AREP to the FME(or AAE)
DTU_ind	AREP	remote_ip_address, data	This is a primitive used to convey a UnconfirmedService indication primitive from the AREP to the FME(or AAE)

10.1.2 Primitive parameters exchanged between AREP and FME(or AAE)

Table 143 describes the parameters used with primitives exchanged between AREP and FME(or AAE).

Table 143 – Primitives parameters exchanged between AREP and FME(or AAE)

Parameter name	Description
remote_ip_address	This parameter transfers 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 transfers the data that the sender will send and the receiver will receive

10.1.3 Primitives Exchanged between AREP and ESME

The primitives exchanged between AREP and ESME are shown in Table 144 and Table 145.

Table 144 – Primitives issued by AREP to ESME

Primitive name	Source	Associated parameters	Functions
Type 14_PDU_req	AREP	remote_ip_address, data	This is a primitive used to convey a ConfirmedService request primitive from the AREP to the ESME

Table 145 – Primitives issued by ESME to AREP

Primitive name	Source	Associated parameters	Functions
Type 14_PDU_ind	ESME	remote_ip_address, data	This is a primitive used to convey a ConfirmedService indication primitive from the ESME to the AREP

10.1.4 Primitive parameters exchanged between AREP and ESME

Table 146 describes the parameters used with primitives exchanged between AAE and ESME.

Table 146 – Primitive parameters exchanged between AREP and ESME

Parameter name	Description
remote_ip_address	This parameter transfers 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 transfers the data that the sender will send and the receiver will receive

10.2 AREP state description

Type 14 Application Relation Endpoint is always active. Its state is described in Table 147.

Table 147 – AREP state descriptions

States	Descriptions
ACTIVE	The AREP in ACTIVE state prepares to transfer service primitives to ALU and ESME, or to receive primitives from ALU and ESME

10.3 State transitions

The protocol state machine of AREP is illustrated in the Figure 9 and Table 148.

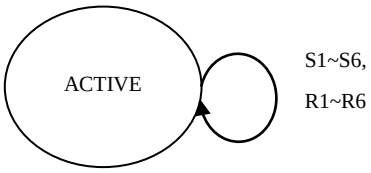


Figure 9 – AREP state transition diagrams

Table 148 – AREP state transitions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	DTC_req
		|| DTC_rsp
		|| DTU_req
		=>
		Type 14_PDU_req {
		user_data := Data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R1	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = “peer”
		&& MessageType(data) = “Confirmed Service Indication”
		=>
		DTC_cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R2	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = “peer”
		&& MessageType(data) = “Confirmed Service Confirmation”
		=>
		DTC_cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

R3	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = “client”
		=>
		DTC_cnf {
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R4	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = “server”
		=>
		DTC_ind{
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

R5	ACTIVE	Type 14_PDU_ind && AREPType(data) = “publisher” => No action	ACTIVE

R6	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = “subscriber”
		=>
		DTU_ind{
		Data := user_data
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

10.4 Function descriptions

Table 149 describes the functions used by AREP state transitions.

Table 149 – AREPType() descriptions

Name	AREPType	Used in	AREP
Input		Output	
Data		The type of AREP	
Function			
To judge the type of AREP, including peer, client, server, publisher and subscriber			

Table 150 describes the functions used by AAE state transitions.

Table 150 – 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 non-confirm service indication primitive			

11 DLL mapping protocol machine

11.1 Concept

In Type 14 system, both UPD/IP and ISO/IEC 8802-3 protocols are applied directly for Type 14 as sublayers of DLL defined in IEC 61158-1. This clause defines the interface between

Type 14 FAL services and UDP/IP layer which is called Type 14 Socket Mapping Entity (ESME).

11.2 Primitives

11.2.1 The primitives and parameters exchanged between AAE and ESME

The primitives exchanged between AAE and ESME are described in 9.1.3.

The parameters used with the primitives between AAE and ESME are described in 9.1.4.

11.2.2 The primitives and parameters exchanged between FME and ESME

The primitives exchanged between FME and ESME are described in 8.1.3.

The parameters used with the primitives exchanged between FME and ESME are described in 8.1.4.

11.2.3 Transport Layer and ESME primitive

Table 151 describes the primitives exchanged between Transport layer (UDP) and ESME.

Table 151 – The primitives exchanged between transport layer and ESME

Primitive name	source	Reference parameters
Type 14_PDU_req	Socket mapping entity	remote_ip_address, data
Type 14_PDU_ind	Transport layer	remote_ip_address, data

11.2.4 Primitives parameters exchanged between Transport Layer and ESME

Table 152 illustrates the primitives parameters exchanged between Transport Layer and ESME.

Table 152 – Primitives parameters exchanged between Transport Layer and ESME

Parameter name	description
remote_ip_address	This parameter transfers 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 transfers the data that the sending side will send and the receiving side will receive

11.3 State description

ESME is always active. Its state is described in Table 153.

Table 153 – ESME state description

State name	description
ACTIVE	ESME transfers primitives to AAE, FME to transport layer; or it is ready to receive primitives from AAE, FME to transport layer

11.4 State transitions

Figure 10 and Table 154 illustrates ESME state transitions.

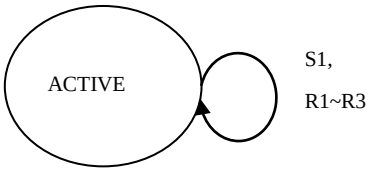


Figure 10 – ESME state transition

Table 154 – ECFME state transitions

#	Current state	Event or condition => action	Next state

	ACTIVE	DTC_req
		|| DTC_rsp
		|| DTU_req
		=>
		Type 14-PDU.req {
		Senddata := data,
		Destination_ip : = remote_ip_address,
		}
			ACTIVE

	ACTIVE	Type 14-PDU.ind
		&& ServiceType(data) = “Confirmed Service Indication”
		=>
		DTC.ind{
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

	ACTIVE	Type 14-PDU.ind
		&& ServiceType(data) = “Confirmed Service Confirmation”
		=>
		DTC.cnf{
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

	ACTIVE	Type 14-PDU.ind
		&& ServiceType(data) = “Unconfirmed Service Indication”
		=>
		DTU.ind{
		Receivedata := data,
		Remote_ip : = remote_ip_address,
		}
			ACTIVE

11.5 Function description

Table 155 describes the function used in ESME state transitions.

Table 155 – ServiceType()description

name	ServiceType	use	Socket mapping entity
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			

Bibliography

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

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

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

ISO/IEC/TR 8802-1, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 1: Overview of Local Area Network Standards*

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

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 8877, *Information technology – Telecommunications and information exchange between systems – Interface connector and contact assignments for ISDN Basic Access Interface located at reference points S and T*

ISO 8601, *Data elements and interchange formats – Information interchange – Representation of dates and times*

IEEE 802.1Q, *IEEE standard for Local and metropolitan area networks – Virtual bridged local area networks*, available at <http://www.ieee.org>

_____

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-14: Spécification du protocole de la couche application – Eléments de type 14

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

L'attention est attirée sur le fait que l'utilisation du type de protocole associé est restreinte par les détenteurs des droits de propriété intellectuelle. En tout état de cause, l'engagement de renonciation partielle aux droits de propriété intellectuelle pris par les détenteurs de ces droits autorise l'utilisation d'un type de protocole de couche avec les autres protocoles de couche du même type, ou dans des combinaisons avec d'autres types autorisées explicitement par les détenteurs des droits de propriété intellectuelle pour ce type.

NOTE Les combinaisons de types de protocole sont spécifiées dans la CEI 61784-1 et la CEI 61784-2.

La Norme internationale CEI 61158-6-14 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette troisième édition annule et remplace la deuxième édition publiée en 2010. Cette édition constitue une révision technique. Les principales modifications par rapport à l'édition précédente sont les suivantes:

- correction des erreurs éditoriales;
- modifications de la spécification de la CPF4;
- mise à jour des exigences de l'ensemble des classes de conformité;
- mise à jour des exigences de l'ensemble des services de conformité.

Le texte de la présente norme est issu des documents suivants:

FDIS	Rapport de vote
65C/764/FDIS	65C/774/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote qui a abouti à l'approbation de la présente norme.

La présente publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Une liste de toutes les parties de la série CEI 61158, publiées sous le titre général *Réseaux de communication industriels – Spécifications des bus de terrain*, peut être consultée sur le site web de la CEI.

Le comité a décidé que le contenu de la présente publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera:

- reconduite;
- supprimée;
- remplacée par une édition révisée, ou
- amendée.

INTRODUCTION

La présente partie de la CEI 61158 s'inscrit dans une série créée pour faciliter l'interconnexion des composants de systèmes d'automation. Elle est relative aux autres normes de l'ensemble défini par le modèle de référence de bus de terrain "à trois couches" décrit dans la CEI 61158-1.

Le protocole d'application fournit le service d'application au moyen des services disponibles au niveau de la couche Liaison de données ou de la couche immédiatement inférieure. Le principal objectif de la présente norme est de définir un ensemble de règles de communication, exprimées en termes de procédures que doivent suivre les entités d'application (Application Entity, AE) homologues au moment de la communication. Ces règles de communication ont pour vocation de fournir une base de développement stable visant à atteindre différents objectifs:

- en tant que guide pour les développeurs et les concepteurs;
- réaliser les essais et acquérir l'équipement;
- dans un accord d'intégration des systèmes dans l'environnement de systèmes ouverts;
- dans le cadre d'une meilleure compréhension des communications à contrainte de temps au sein de l'OSI.

La présente norme porte en particulier sur la communication et l'interfonctionnement des capteurs, des effecteurs et d'autres appareils d'automatisation. Grâce à cette norme associée à d'autres normes des modèles de référence OSI ou de bus de terrain, des systèmes par ailleurs incompatibles peuvent fonctionner ensemble, quelle que soit leur combinaison.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-14: Spécification du protocole de la couche application – Eléments de type 14

1 Domaine d'application

1.1 Généralités

La couche application de bus de terrain (Fieldbus Application Layer, FAL) procure aux programmes de l'utilisateur un moyen d'accès à l'environnement de communication des bus de terrain. A cet égard, la FAL peut être considérée comme une "fenêtre entre programmes d'application correspondants".

La présente norme fournit des éléments communs pour les communications de messagerie en temps critique ou non entre des programmes d'application dans un environnement et avec un matériel d'automation spécifiques aux bus de terrain de type 14. Le terme "à temps critique" sert à représenter la présence d'une fenêtre temporelle dans les limites de laquelle une ou plusieurs actions spécifiées sont exigées d'être parachevées avec un certain niveau défini de certitude. Le manquement à parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent ces actions, avec le risque concomitant pour l'équipement, les installations et éventuellement pour la vie humaine.

La présente norme spécifie les interactions entre les applications distantes et définit le comportement, visible par un observateur externe, assuré par la couche application de bus de terrain de type 14, en termes

a) de syntaxe abstraite formelle définissant les unités de données de protocole de couche application, acheminées entre les entités d'application en communication;

b) de syntaxe de transfert définissant les règles de codage qui s'appliquent aux unités de données de protocole de couche application;

c) du diagramme d'états de contexte application définissant le comportement de service application visible entre des entités d'application engagées dans une communication;

d) de diagrammes d'états de relations d'applications définissant le comportement de communication visible entre les entités d'application en communication.

La présente norme vise à définir le protocole mis en place pour

a) définir la représentation filaire des primitives de service définies dans la CEI 61158-5-14 et

b) définir les caractéristiques visibles en externe liées à leur transfert.

La présente norme spécifie le protocole de la couche application de bus de terrain de type 14, en conformité avec le modèle de référence de base OSI (ISO/CEI 7498) et avec la structure de la couche application OSI (ISO/CEI 9545).

1.2 Spécifications

La présente norme a pour principal objectif de préciser la syntaxe et les caractéristiques du protocole de couche application qui transmet les services de couche application définis dans la CEI 61158-5-14.

Un objectif secondaire est de fournir des trajets de migration à partir de protocoles de communications industrielles préexistants. Ce dernier objectif explique la diversité des protocoles normalisés dans la série CEI 61158-6.

1.3 Conformité

La présente norme ne définit pas de mises en œuvre ni de produits particuliers, pas plus qu'elle ne limite les mises en œuvre des entités de couche application dans les systèmes d'automatisation industriels. La conformité est obtenue par le biais de la mise en œuvre de cette spécification de protocoles de couche application.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

NOTE Toutes les parties de la série CEI 61158, ainsi que la CEI 61784-1 et la CEI 61784-2 font l'objet d'une maintenance simultanée. Les références croisées à ces documents dans le texte se rapportent par conséquent aux éditions datées dans la présente liste de références normatives.

CEI 61158-3-14, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-14: Définition des services de la couche liaison de données – Eléments de type 14*

CEI 61158-4-14, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-14: Spécification du protocole de la couche liaison de données – Eléments de type 14*

CEI 61158-5-14, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-14: Définition des services de la couche application – Eléments de type 14*

CEI 61158-6 (toues les parties), *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6: Spécification du protocole de la couche application*

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange* (disponible en anglais seulement)

ISO/IEC 2375, *Information technology – Procedure for registration of escape sequences and coded character sets* (disponible en anglais seulement)

ISO/CEI 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base – Partie 1: Le modèle de base*

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

ISO/CEI 8822, *Technologies de l'information – Interconnexion de systèmes ouverts – Définition du service de présentation*

ISO/CEI 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de la notation de syntaxe abstraite numéro 1 (ASN.1)*[1]

_____

[1] Retirée.

ISO/CEI 9545, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Structure de la couche Application*

ISO/CEI 10731, *Technologies de l'information – Interconnexion de systèmes ouverts – Modèle de Référence de Base – Conventions pour la définition des services OSI*

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic* (disponible en anglais seulement)

IEEE 754-2008, *IEEE Standard for Floating-Point Arithmetic*

3 Termes, définitions, symboles, abréviations et conventions

Pour les besoins du présent document, les termes, définitions, symboles, abréviations et conventions suivants s'appliquent.

3.1 Termes et définitions référencés

3.1.1 Termes de l'ISO/CEI 7498-1

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 7498-1 s'appliquent:

a) entité d'application
b) processus d'application
c) unité de données de protocole d'application
d) élément de service d'application
e) invocation d'entités d'application
f) invocation de processus d'application
g) transaction d'applications
h) système ouvert réel
i) syntaxe de transfert

3.1.2 Termes de l'ISO/CEI 8822

Pour les besoins du présent document, les termes suivants définis dans l'ISO/CEI 8822 s'appliquent:

a) syntaxe abstraite
b) contexte de présentation

3.1.3 Termes de l'ISO/CEI 9545

Pour les besoins du présent document, les termes suivants définis dans l'ISO/CEI 9545 s'appliquent:

a) association d'applications
b) contexte d'application
c) nom de contexte d'application
d) invocation d'entités d'application
e) type d'entité d'application
f) invocation de processus d'application
g) type de processus d'application

h) élément de service d'application

i) élément de service de contrôle d'application

3.1.4 Termes de l'ISO/CEI 8824

Pour les besoins du présent document, les termes suivants définis dans l'ISO/CEI 8824 s'appliquent:

a) identificateur d'objet

b) type

3.1.5 Termes relatifs à la couche Liaison de données de bus de terrain

Pour les besoins du présent document, les termes suivants, définis dans les normes CEI 61158-3-14 et CEI 61158-4-14, s'appliquent.

a) DL-Time (temps de liaison de données)

b) DL-Scheduling-Policy (politique de programmation de liaison de données)

c) DLCEP

d) DLC

e) mode orienté connexion DL

f) DLPDU

g) DLSDU

h) DLSAP

i) liaison

j) adresse réseau

k) adresse de nœud

l) nœud

m) programmé

3.2 Termes et définitions relatifs à la couche application de bus de terrain

3.2.1
contrôle d'accès
contrôle des opérations de lecture et d'écriture portant sur un objet

3.2.2
chemin d'accès
association d'un nom symbolique avec une variable dans le contexte d'une communication ouverte

3.2.3
macrocycle de communication
ensemble de cycles élémentaires nécessaires aux activités de communication configurées dans un macro-segment réseau

3.2.4
programmation de la communication
algorithmes et comportement des opérations de transfert de données se déroulant de manière déterministe et reproductible

3.2.5
configuration (d'un système ou appareil)
étape de la conception d'un système consistant en la sélection des unités fonctionnelles, l'attribution de leurs emplacements et la définition de leurs interconnexions

3.2.6
cyclique
répétitif d'une manière régulière

3.2.7
instance du bloc de fonctions de destination
instance du bloc de fonctions qui reçoit les paramètres spécifiés

3.2.8
domaine
partie de la mémoire destinée au stockage de code ou de données

3.2.9
téléchargement de domaine
opération consistant à écrire des données dans un domaine

3.2.10
chargement de domaine
opération consistant à lire des données à partir d'un domaine

3.2.11
entité
élément particulier (personne, lieu, processus, objet, concept, association ou événement)

3.2.12
Pont de type 14
entité de relais DL qui effectue des synchronisations entre les liaisons (bus) et peut assurer des fonctions de stockage-transmission et d'acheminement sélectives dans le but de raccorder deux micro-segments réseau

3.2.13
identificateur
mot de 16 bits associé à une variable système

3.2.14
index
adresse d'un objet au sein d'un processus d'application

3.2.15
instance
occurrence physique permettant d'identifier un objet parmi d'autres au sein d'une même classe d'objets

3.2.16
instanciation
création d'une instance d'un type spécifié

3.2.17
informations de gestion
informations mises à disposition sur le réseau pour les fins de gestion du système sur site

3.2.18
base d'informations de gestion
liste organisée d'informations de gestion

3.2.19
mapping
ensemble de valeurs ayant une correspondance définie avec les grandeurs ou valeurs d'un autre ensemble

3.2.20
membre
élément d'un attribut qui est structuré comme étant un élément d'une matrice

3.2.21
filtrage de messages
décision relative à un message conformément à une règle spéciale

3.2.22
micro-segment
partie d'un réseau où une programmation spéciale est mise en œuvre

3.2.23
position relative
nombre d'octets à partir d'une position désignée spécialement

3.2.24
phase
partie écoulée d'un cycle, mesurée à partir d'un point d'origine fixe

3.2.25
interface de processus
échange de données et mapping d'informations entre un processus physique et une unité d'application

3.2.26
temps réel
capacité d'un système à fournir un résultat exigé dans un délai limité

3.2.27
communication en temps réel
transfert de données temps réel

3.2.28
Real-Time Ethernet
RTE
réseau conforme à l'ISO/CEI 8802-3 qui inclut la communication en temps réel

Note 1 à l'article: D'autres formes de communication peuvent être prises en charge, sous réserve que la communication en temps réel ne soit pas compromise.

Note 2 à l'article: Cette définition s'applique, mais sans que cela soit limitatif, à l'ISO/CEI 8802-3. Elle peut s'appliquer à d'autres spécifications IEEE 802, par exemple IEEE 802.1Q.

3.2.29
programmation
configuration temporelle d'un certain nombre d'opérations connexes

3.2.30
macrocycle de programmation
intervalle de temps pour mettre en œuvre un programme spécifique

3.2.31
instance FB source
instance de FB qui envoie un paramètre spécifique

3.2.32
décalage temporel
laps de temps écoulé à partir d'une heure précise

3.3 Abréviations et symboles

AAE	Application Access Entity (entité d'accès à l'application)
AE	Application Entity (entité d'application)
AL	Application Layer (couche application)
ALE	Entité de la couche application
ALP	Application Layer Protocol (protocole de couche application)
APO	Application Object (objet d'application)
AP	Application Process (processus d'application)
APDU	Application Protocol Data Unit (unité de donnée de protocole application)
API	Application Process Identifier (identificateur de processus d'application)
AR	Application Relationship (relation d'application)
ARP	Address Resolution Protocol (protocole de résolution d'adresse)
AREP	Application Relationship End Point (point terminal de relation d'application)
ASE	Application Service Element (élément de service application)
Cnf	Confirmation
CR	Communication Relationship (relation de communication)
CREP	Communication Relationship End Point (point d'extrémité de relation de communication)
CSMA/CD	Carrier Sense Multiple Access with Collision Detection (accès multiple par surveillance du signal et détection de collision)
DD	Device Description (description d'appareil)
DHCP	Dynamic Host Configuration Protocol (protocole de configuration d'hôte dynamique)
DL-	(comme préfixe) data-link- (liaison de données)
DLCEP	Data-link Connection End Point (point d'extrémité de connexion de couche de liaison de données)
DLL	Couche Liaison de données
DLE	Entité de la couche Liaison de données
DLM	Data-Link-Management (gestion de liaison de données)
DLS	Service de la couche Liaison de données
DLSAP	Point d'accès de service de la couche Liaison de données
DLSDU	DL-Service-Data-Unit (unité de données de service DL)
ECSME	Entité de gestion de la programmation des communications de type 14
EM_	(comme préfixe) Gestion de type 14
ESME	Entité de mapping de ports de type 14
FB	Function Block (bloc de fonction)
FBAP	Function Block Application Process (processus d'applications du bloc de fonctions)
FME	Entité de gestion de la couche application de bus de terrain
FRT	Fast Real-time (rapide en temps réel)
Ind	Indication
IP	Internet Protocol (protocole Internet)
LLC	Logical Link Control (contrôle de liaison logique)
LMP	Link Management Protocol (protocole de gestion de liaison)

MAC	Medium Access Control (commande d'accès au support)
MAU	Medium Attachment Unit (unité de raccordement au support)
MOB	Management Object Base (base d'objets de gestion)
PAD	Contact (bits)
PDU	Protocol Data Unit (unité de données de protocole)
P/S	Publisher/Subscriber (publication/abonnement)
Req	Request (demande)
Rsp	Response (réponse)
RTE	Real-Time Ethernet (Ethernet en temps réel)
RT-Ethernet	Real-Time Ethernet (Ethernet en temps réel)
SAP	Service Access Point (point d'accès au service)
SDU	Service Data Unit (unité de données de service)
SME	System Management Entity (entité de gestion système)
SNTP	Simple Network Time Protocol (protocole simple d'heure réseau)
TCP	Transmission Control Protocol (protocole de commande de transport)
UDP	User Datagram Protocol (protocole datagramme d'utilisateur)
.cnf	Primitive de confirmation
.ind	Primitive d'indication
.req	Primitive de demande
.rsp	Primitive de réponse

3.4 Conventions

3.4.1 Concept général

La couche FAL est définie comme un ensemble d'éléments ASE orientés objet. Chaque ASE est spécifié dans un paragraphe distinct. Chaque spécification d'ASE est constituée de trois parties: ses définitions de classe, ses services et sa spécification de protocole. Les deux premiers éléments sont contenus dans la CEI 61158-5-14. La spécification des protocoles pour chacun des éléments ASE est définie dans la présente norme.

Les définitions de classe définissent les attributs des classes prises en charge par chaque élément ASE. Les attributs sont accessibles à partir des instances de la classe qui utilise les services ASE de gestion spécifiés dans la CEI 61158-5-14. La spécification de services définit les services qui sont fournis par l'ASE.

La présente norme emploie les conventions de description énoncées dans l'ISO/CEI 10731.

3.4.2 Conventions relatives aux diagrammes d'états pour les éléments de type 14

Un diagramme d'états décrit la séquence d'états par lesquels passe une entité; il peut se matérialiser par un diagramme de passages d'état et/ou une table d'états.

Dans un diagramme de passages d'états (Figure 1), les passages d'états sont représentés par des cercles et sont illustrés par une flèche à côté de laquelle sont indiqués les événements ou conditions sous-jacents.

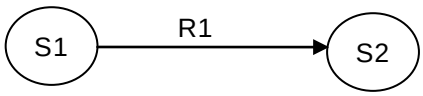


Figure 1 – Diagramme de passages d'état

Tableau 1 – Eléments de la description d'un diagramme d'états

#	Etat actuel	Evénements ou conditions déclenchant ce passage d'état => Action	Etat suivant
Nom de ce passage	Etat actuel auquel s'applique ce passage d'état	Evénements ou conditions déclenchant ce passage d'état. => Actions effectuées lorsque les événements ou conditions ci-dessus sont réunis. Elles figurent toujours au-dessous des événements ou conditions et sont toujours présentées avec un retrait	Etat suivant dans lequel passe le diagramme d'états une fois les actions effectuées

Les conventions utilisées dans la table des passages d'état (Tableau 1) sont les suivantes.

:= La valeur d'un élément, à gauche, est remplacée par la valeur d'un élément, à droite. Si un élément à droite est un paramètre, il provient de la primitive indiquée comme événement d'entrée.

xxx indique un nom de paramètre.

Exemple:

Identifier:= reason
signifie que la valeur d'un paramètre "reason" est attribuée à un paramètre appelé "Identifier".
"xxx" indique une valeur fixe.

Exemple:

Identifier:= "abc"
signifie que la valeur "abc" est attribuée à un paramètre appelé "Identifier".

= Condition logique indiquant qu'un élément à gauche est égal à un élément à droite.
< Condition logique indiquant qu'un élément à gauche est inférieur à l'élément à droite.
> Condition logique indiquant qu'un élément à gauche est supérieur à l'élément à droite.
<> Condition logique indiquant qu'un élément à gauche est différent d'un élément à droite.
&& "ET" Logique
|| "OU" Logique

Service.req représente une primitive de demande; Service.req{} signifie qu'une primitive de demande est envoyée;

Service.ind représente une primitive d'indication; Service.ind{} signifie qu'une primitive d'indication est reçue;

Service.rsp représente une primitive de réponse; Service.rsp{} signifie qu'une primitive de réponse est envoyée;

Service.cnf représente une primitive de confirmation; Service.cnf{} signifie qu'une primitive de confirmation est reçue.

4 Syntaxe abstraite

4.1 Description des unités PDU de format fixe

Les unités PDU de type 14 se composent d'un en-tête de longueur fixe et d'un corps de longueur variable. L'en-tête de l'unité PDU contient le type de service, le type et la longueur du message, etc.

```
Type 14 PDU: : = CHOICE {
        confirmed-RequestPDU            [0]     IMPLICIT Confirmed-RequestPDU,
        confirmed-ResponsePDU           [1]     IMPLICIT Confirmed-ResponsePDU,
        confirmed-ErrorPDU              [2]     IMPLICIT Confirmed-ErrorPDU,
        unconfirmed-RequestPDU          [3]     IMPLICIT Unconfirmed-RequestPDU
}
Confirmed-RequestPDU: : = SEQUENCE {
        pduHeader                       PDUHeader,
        confirmed-request               Confirmed-Request
}
Confirmed-ResponsePDU: : = SEQUENCE {
        pduHeader                       PDUHeader,
        confirmed-response              Confirmed-Response
}
Confirmed-ErrorPDU: : = SEQUENCE {
        pduHeader                       PDUHeader,
        confirmed-error                 Confirmed-Error
}
Unconfirmed-RequestPDU: : = SEQUENCE {
        pduHeader                       PDUHeader,
        unconfirmed-request             Unconfirmed-Request
}
```

4.1.1 Service Confirmed Request

```
Confirmed- Request: : = CHOICE {
        EM_GetDeviceAttribute           [0]     IMPLICIT EM_GetDeviceAttribute-RequestPDU,
        EM_ConfiguringDevice            [1]     IMPLICIT EM_ConfiguringDevice-RequestPDU,
        EM_SetDefaultValue              [2]     IMPLICIT EM_SetDefaultValue-RequestPDU,
        DomainDownload                  [3]     IMPLICIT DomainDownload-RequestPDU,
        DomainUpload                    [4]     IMPLICIT DomainUpload-RequestPDU,
        AcknowledgeEventRoport          [5]     IMPLICIT AcknowledgeEventNotifi-RequestPDU,
        ReportConditionChanging         [6]     IMPLICIT AlterEventConditionMon-RequestPDU,
        Read                            [7]     IMPLICIT Read-RequestPDU,
        Write                           [8]     IMPLICIT Write-RequesPDU,
        FRTRead                         [9]     IMPLICIT FRTRead-RequestPDU,
        FRTWrite                        [10]    IMPLICIT FRTWrite-RequesPDU
        BlockTransmissionOpen           [11]    IMPLICIT OpenBlockTransmission-RequestPDU,
        BlockTransmissionClose          [12]    IMPLICIT CloseBlockTransmission-RequestPDU,
}
```

4.1.2 Service Confirmed Response

```
Confirmed- Response: : = CHOICE {
        EM_GetDeviceAttribute           [0] IMPLICIT EM_GetDeviceAttribute-ResponsePDU,
        EM_ConfiguringDevice            [1] IMPLICIT EM_ConfiguringDevice-ResponsePDU,
        EM_SetDefaultValue              [2] IMPLICIT EM_SetDefaultValue-ResponsePDU,
        DomainDownload                  [3] IMPLICIT DomainDownload-ResponsePDU,
        DomainUpload                    [4] IMPLICIT DomainUpload-ResponsePDU,
        AcknowledgeEventRoport          [5] IMPLICIT AcknowledgeEventNotifi-ResponsePDU,
        ReportConditionChanging         [6] IMPLICIT AlterEventConditionMon-ResponsePDU,
        Read                            [7] IMPLICIT Read-ResponsePDU,
        Write                           [8] IMPLICIT Write-ResponsePDU,
        FRTRead                         [9] IMPLICIT FRTRead-ResponsePDU,
        FRTWrite                        [10] IMPLICIT FRTWrite-ResponsePDU
```

```
        BlockTransmissionOpen     [11] IMPLICIT OpenBlockTransmission-ResponsePDU,
        BlockTransmissionClose    [12] IMPLICIT CloseBlockTransmission-ResponsePDU,
}
```

4.1.3 Confirmed Error

```
Confirmed- Error: : = CHOICE {
        EM_GetDeviceAttribute     [0]    IMPLICIT Error-Type,
        EM_ConfiguringDevice      [1]    IMPLICIT Error-Type,
        EM_SetDefaultValue        [2]    IMPLICIT Error-Type,
        DomainDownload            [3]    IMPLICIT Error-Type,
        DomainUpload              [4]    IMPLICIT Error-Type,
        AcknowledgeEventRoport    [5]    IMPLICIT Error-Type,
        ReportConditionChanging   [6]    IMPLICIT Error-Type,
        Read                      [7]    IMPLICIT Error-Type,
        Write                     [8]    IMPLICIT Error-Type,
        FRTRead                   [9]    IMPLICIT Error-Type,
        FRTWrite                  [10]   IMPLICIT Error-Type
        BlockTransmissionOpen     [11]   IMPLICIT Error-Type
        BlockTransmissionClose    [12]   IMPLICIT Error-Type
}
```

4.1.4 Error type

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

4.1.5 Classe d'erreurs

```
ErrorClass::= CHOICE {                                          ErrorCode
    Ressource                 [0]   IMPLICIT Integer8 {
                                        memory-unavailable      (0),
                                        Autre                   (1)
    },
    Service                   [1]   IMPLICIT Integer8 {
                                        object-state-conflict   (0),
                                        object-constraint-conflict (1),
                                        parameter-inconsistent  (2),
                                        illegal-parameter       (3),
                                        Size Error              (4),
                                        Autre                   (5)
    }
    Accès                     [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),
                                        Autre                   (7)
    }
    Temporisateur             [3]   IMPLICIT Integer8 {
                                        Timer-Expire            (0),
                                        Timer-Error             (1),
                                        Autre                   (2)
    },
    Autre                     [4]   IMPLICIT Integer8 {
                                        Autre                   (0)
    }
}
```

4.1.6 Unconfirmed Request

```
Unconfirmed-Request: : = CHOICE {
        EM_DetectingDevice        [0]    IMPLICIT EM_DetectingDevice-RequestPDU,
```

```
        EM_OnlineReply              [1]     IMPLICIT EM_OnlineReply-RequestPDU,
        EM_ActiveNotification       [2]     IMPLICIT EM_ActiveNotification-RequestPDU,
        EventRoport                 [3]     IMPLICIT EventRoport-RequestPDU,
        VariableDistribute          [4]     IMPLICIT VariableDistribute-RequestPDU,
        FRTVariableDistribute       [5]     IMPLICIT FRTVariableDistribute-RequestPDU
        BlockTransmit               [6]     IMPLICIT BlockTransmit-RequestPDU
        BlockTransmissionHeartbeat[7]       IMPLICIT RequestPDU-RequestPDU
}
```

4.1.7 Application layer PDU Type 14

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

4.1.8 Format d'en-tête PDU

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

4.1.9 FAL Management Entity services

4.1.9.1 Service EM_DetectingDevice

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

4.1.9.2 Service EM_OnlineReply

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

4.1.9.3 Service EM_GetDeviceAttribute

```
EM_GetDeviceAttribute-RequestPDU: : = SEQUENCE {
        DestinationIPAddress        [0]     IMPLICIT Unsigned32,
}
EM_GetDeviceAttribute-ResponsePDU: : = CHOICE {
        EM_GetDeviceAttribute-PositiveResponsePDU,
        EM_GetDeviceAttribute-NegativeResponsePDU
}
EM_GetDeviceAttribute-PositiveResponsePDU: : = SEQUENCE {
        DeviceID                    [0]     IMPLICIT VisibleString,
        PdTag                       [1]     IMPLICIT VisibleString,
        Status                      [2]     IMPLICIT Unsigned8,
        DeviceType          [3]     IMPLICIT Unsigned8,
```

```
        Annunciation Interval                [4]    IMPLICIT Unsigned16,
        Annunciation Version Number          [5]    IMPLICIT Unsigned16,
        DuplicateTagDetected                 [6]    IMPLICIT Boolean,
        DeviceRedundancyNumber               [7]    IMPLICIT Unsigned8,
        LANRedundancyPort                    [8]    IMPLICIT Unsigned16
        DeviceRedundancy State               [9]    IMPLICIT Unsigned8,
        MaxRedundancyNumber                  [10]   IMPLICIT Unsigned8,
        ActiveIPAddress                      [11]   IMPLICIT Unsigned32
}
EM_GetDeviceAttribute-NegativeResponsePDU: : = SEQUENCE {
        DestinationIPAddress                 [0]    IMPLICIT    Unsigned32,
        Error-Type                           [1]    IMPLICIT    Error-Type
}
```

4.1.9.4 Service EM_ActiveNotification

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

4.1.9.5 Service EM_ConfiguringDevice

```
EM_ConfiguringDevice-RequestPDU: : = SEQUENCE {
        DestinationIPAddress                 [0]    IMPLICIT Unsigned32,
        DeviceID                             [1]    IMPLICIT VisibleString,
        PdTag                                [2]    IMPLICIT VisibleString,
        AnnunciationInterval                 [3]    IMPLICIT Unsigned16,
        DuplicateTagDetected                 [4]    IMPLICIT Boolean,
        DeviceRedundancyNumber               [5]    IMPLICIT Unsigned8,
        LANRedunancyPort                     [6]    IMPLICIT Unsigned16,
        DeviceRedundancyState                [7]    IMPLICIT Unsigned8,
        MaxRedundancyNumber                  [8]    IMPLICIT Unsigned8,
        ActiveIPAddress                      [9]    IMPLICIT Unsigned32
}
EM_ConfiguringDevice-ResponsePDU: : = CHOICE {
        EM_ConfiguringDevice-PositiveResponsePDU,
        EM_ConfiguringDevice-NegativeResponsePDU
}
EM_ConfiguringDevice-PositiveResponsePDU: : = SEQUENCE {
        DestinationIPAddres          [0]    IMPLICIT Unsigned32,
        MaxRedundancyNumber          [1]    IMPLICIT Unsigned8
}
EM_ConfiguringDevice-NegativeResponsePDU: : = SEQUENCE {
        DestinationIPAddress         [0]    IMPLICIT Unsigned32,
        ErrorType                    [1]    IMPLICIT ErrorType
}
```

4.1.9.6 Service EM_SetDefaultValue

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

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

4.1.10 Services Application Access Entity (AAE)

4.1.10.1 Service DomainDownload

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

4.1.10.2 Service DomainUpload

```
DomainUpload-RequestPDU: : = SEQUENCE {
        SourceAppID                 [0] IMPLICIT Unsigned16,
        DestinationAppID            [1] IMPLICIT Unsigned16,
        DestinationObjectID         [2] IMPLICIT Unsigned16,
        DataNumber                  [3] IMPLICIT Unsigned16
}
DomainUpload-ResponsePDU: : = CHOICE {
        DomainUpload-PositiveResponsePDU,
        DomainUpload-NegativeResponsePDU
}
DomainUpload-PositiveResponsePDU: : = SEQUENCE {
        DestinationAppID            [0] IMPLICIT Unsigned16,
        DataLength                  [1] IMPLICIT Unsigned16,
        MoreFollows                 [2] IMPLICIT Boolean,
        Reserved                    [3] IMPLICIT OctetString,
        LoadData                    [4] IMPLICIT OctetString
}
DomainUpload-NegativeResponsePDU: : = SEQUENCE {
        DestinationAppID            [0]     IMPLICIT Unsigned16,
        Reserved                    [1]     IMPLICIT OctetString,
        ErrorType                   [2]     IMPLICIT ErrorType
}
```

4.1.10.3 Service EventRoport

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

4.1.10.4 Service AcknowledgeEventRoport

```
AcknowledgeEventRoport-RequestPDU: : = SEQUENCE {
        DestinationAppID        [0] IMPLICIT Unsigned16,
        DestinationObjectID     [1] IMPLICIT Unsigned16,
        EventNumber             [2] IMPLICIT Unsigned16
}
AcknowledgeEventRoport -ResponsePDU: : = CHOICE {
        AcknowledgeEventRoport -PositiveResponsePDU,
        AcknowledgeEventRoport -NegativeResponsePDU
}
AcknowledgeEventRoport -PositiveResponsePDU: : = SEQUENCE{
        DestinationAppID        [0] IMPLICIT Unsigned16
}
AcknowledgeEventRoport -NegativeResponsePDU: : = SEQUENCE{
        DestinationAppID        [0] IMPLICIT Unsigned16
        Reserved                [1] IMPLICIT OctetString,
        ErrorType               [2] IMPLICIT ErrorType
}
```

4.1.10.5 Service ReportConditionChanging

```
ReportConditionChanging-RequestPDU: : = SEQUENCE {
        DestinationAppID        [0] IMPLICIT Unsigned16,
        DestinationObjectID     [1] IMPLICIT Unsigned16,
        Enabled                 [2] IMPLICIT Boolean
}
ReportConditionChanging -ResponsePDU: : = CHOICE {
        ReportConditionChanging -PositiveResponsePDU,
        ReportConditionChanging -NegativeResponsePDU
}
ReportConditionChanging -PositiveResponsePDU: : = SEQUENCE{
        DestinationAppID        [0] IMPLICIT Unsigned16
}
ReportConditionChanging -NegativeResponsePDU: : = SEQUENCE{
        DestinationAppID        [0] IMPLICIT Unsigned16
        Reserved                [1] IMPLICIT OctetString,
        ErrorType               [2] IMPLICIT ErrorType
}
```

4.1.10.6 Service Read

```
Read-RequestPDU: : = SEQUENCE {
        DestinationAppID        [0] IMPLICIT Unsigned16,
        DestinationObjectID     [1] IMPLICIT Unsigned16,
        SubIndex                [2] IMPLICIT Unsigned16
}
Read -ResponsePDU: : = CHOICE {
        Read -PositiveResponsePDU,
        Read -NegativeResponsePDU
}
Read-PositiveResponsePDU: : = SEQUENCE {
        DestinationAppID        [0] IMPLICIT Unsigned16,
        Reserved                [1] IMPLICIT OctetString,
```

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

4.1.10.7 Write service

```
Write-RequestPDU: : = SEQUENCE {
        DestinationAppID        [0] IMPLICIT Unsigned16,
        DestinationObjectID     [1] IMPLICIT Unsigned16,
        SubIndex                [2] IMPLICIT Unsigned16,
        Reserved                [3] IMPLICIT OctetString,
        Data                    [4] IMPLICIT OctetString
}
Write -ResponsePDU: : = CHOICE {
        Write -PositiveResponsePDU,
        Write -NegativeResponsePDU
}
Write -PositiveResponsePDU: : = SEQUENCE {
        DestinationAppID        [0] IMPLICIT Unsigned16,
}
Write -NegativeResponsePDU: : = SEQUENCE {
        DestinationAppID        [0] IMPLICIT Unsigned16,
        Reserved                [1] IMPLICIT OctetString,
        ErrorType               [2] IMPLICIT ErrorType
}
```

4.1.10.8 Service VariableDistribute

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

4.1.10.9 Service FRTRead

```
FRTRead-RequestPDU : : = SEQUENCE {
        DestinationObjectID     [0] IMPLICIT Unsigned16,
        SubIndex                [1] IMPLICIT Unsigned16
}
FRTRead -ResponsePDU : : = CHOICE {
        FRTRead -PositiveResponsePDU,
        FRTRead -NegativeResponsePDU
}
FRTRead-PositiveResponsePDU : : = SEQUENCE {
        FRTData                 [0] IMPLICIT OctetString
}
FRTRead-NegativeResponsePDU : : = SEQUENCE {
        ErrorType               [0] IMPLICIT ErrorType
}
```

4.1.10.10 FRTWrite service

```
FRTWrite-RequestPDU : : = SEQUENCE {
        DestinationObjectID     [0] IMPLICIT Unsigned16,
        SubIndex                [1] IMPLICIT Unsigned16,
        Reserved                [2] IMPLICIT OctetString,
        Data                    [3] IMPLICIT OctetString
}
FRTWrite -ResponsePDU : : = CHOICE {
        FRTWrite -PositiveResponsePDU,
```

```
        FRTWrite -NegativeResponsePDU
}
FRTWrite -PositiveResponsePDU : : = SEQUENCE {
        DestinationObjectID  [0] IMPLICIT Unsigned16,
}
FRTWrite -NegativeResponsePDU : : = SEQUENCE {
        ErrorType                [0] IMPLICIT ErrorType
}
```

4.1.10.11 Service FRTVariableDistribute

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

4.1.10.12 Service BlockTransmissionOpen

```
BlockTransmissionOpen-RequestPDU : : = SEQUENCE {
        SourceAppID              [0] IMPLICIT Unsigned16,
        DestinationAppID         [1] IMPLICIT Unsigned16
        DestinationObjectID      [2] IMPLICIT Unsigned16
        BlockType                [3] IMPLICIT Unsigned16
        BlockConfigInfo          [4] IMPLICIT OctetString
}
BlockTransmissionOpen-ResponsePDU : : = CHOICE {
        BlockTransmissionOpen -PositiveResponsePDU,
        BlockTransmissionOpen -NegativeResponsePDU
}
BlockTransmissionOpen-PositiveResponsePDU : : = SEQUENCE {
        DestinationAppID         [0] IMPLICIT Unsigned16,
        Reserved                 [1] IMPLICIT OctetString,
        MultiIPAddress           [2] IMPLICIT Unsigned32
}
BlockTransmissionOpen-NegativeResponsePDU : : = SEQUENCE {
        DestinationAppID         [0] IMPLICIT Unsigned16,
        Reserved                 [1] IMPLICIT OctetString,
        ErrorType                [2] IMPLICIT ErrorType
}
```

4.1.10.13 Service BlockTransmissionClose

```
BlockTransmissionClose-RequestPDU : : = SEQUENCE {
        SourceAppID              [0] IMPLICIT Unsigned16,
        DestinationAppID         [1] IMPLICIT Unsigned16
        DestinationObjectID      [2] IMPLICIT Unsigned16
        BlockType                [3] IMPLICIT Unsigned16
}
BlockTransmissionClose-ResponsePDU : : = CHOICE {
        BlockTransmissionClose -PositiveResponsePDU,
        BlockTransmissionClose -NegativeResponsePDU
}
BlockTransmissionClose-PositiveResponsePDU : : = SEQUENCE {
        DestinationAppID         [0] IMPLICIT Unsigned16,
}
BlockTransmissionClose-NegativeResponsePDU : : = SEQUENCE {
        DestinationAppID         [0] IMPLICIT Unsigned16,
        Reserved                 [1] IMPLICIT OctetString,
        ErrorType                [2] IMPLICIT ErrorType
}
```

4.1.10.14 Service BlockTransmit

```
BlockTransmit-RequestPDU : : = SEQUENCE {
        SourceAppID              [0] IMPLICIT Unsigned16,
        DestinationAppID         [1] IMPLICIT Unsigned16,
```

```
        DestinationObjectID     [2] IMPLICIT Unsigned16,
        DataLength              [3] IMPLICIT Unsigned16,
        BlockType               [4] IMPLICIT Unsigned16
        SequenceNumber          [5] IMPLICIT Unsigned16
        TimeStamp               [6] IMPLICIT PrecisionTimeDifference
        SendCount               [7] IMPLICIT Unsigned16
        BlockData               [8] IMPLICIT OctetString
}
```

4.1.10.15 Service BlockTransmissionHeartbeat

```
BlockTransmissionHeartbeat-RequestPDU : : = SEQUENCE {
        SourceAppID             [0] IMPLICIT Unsigned16,
        DestinationAppID        [1] IMPLICIT Unsigned16,
        DestinationObjectID     [2] IMPLICIT Unsigned16,
        ReceptionCount          [3] IMPLICIT Unsigned16,
        CumulativeLost          [4] IMPLICIT Unsigned16
        Jitter                  [5] IMPLICIT PrecisionTimeDifference
}
```

4.1.11 Syntaxe abstraite du data type

4.1.11.1 Notation du type Boolean

```
        Boolean ::= BOOLEAN             --value is non-zero means TRUE
                                        --la valeur zéro signifie FALSE
```

4.1.11.2 Notation du type Integer

```
        Int8 ::= INTEGER (-128..+127)           -- integer range -27<= i <= 27-1
        Int16 ::= INTEGER (-32 768..+32 767)    -- integer range -215<= i <= 215-1
        Int32 ::= INTEGER                       -- integer range -231<= i <= 231-1
        Int64 ::= INTEGER                       -- integer range -263<= i <= 263-1
```

4.1.11.3 Notation du type Unsigned Integer

```
        Unsigned8 ::= INTEGER (0..255)          -- integer range 0 <= i <= 28-1
        Unsigned16 ::= INTEGER (0..65 535)      -- integer range 0 <= i <= 216-1
        Unsigned32 ::= INTEGER                  -- integer range 0 <= i <= 232-1
        Unsigned64 ::= INTEGER                  -- integer range 0 <= i <= 264-1
```

4.1.11.4 Notation du type Float Data

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

4.1.11.5 Notation du type VisibleString

```
        VisibleString ::= VISIBLE STRING        --general use
```

4.1.11.6 Notation du type OctetString

```
        OctetString ::= Octet STRING            --general use
```

4.1.11.7 Notation du type BitString

```
        BitString ::= BIT STRING                -- general use
```

4.1.11.8 Notation du type TimeofDay

```
        TimeOfDay::= OctetString6
```

4.1.11.9 Notation du type BinaryDate

```
        BinaryDate::= OctetString8
```

4.2 Définitions d'objet dans les éléments ASE de gestion de couches FAL

4.2.1 Objet d'en-tête MOB de type 14

La fonction de l'objet d'en-tête MOB de type 14 est expliquée au Tableau 2.

Tableau 2 – Définition de l'objet d'en-tête MOB de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet d'en-tête MOB de type 14 dans la MOB de type 14
2	MOB Revision Number	Lecture seule	Unsigned16	2	2	Version de la MOB de type 14

4.2.2 Objet de descripteur d'appareil de type 14

La fonction de l'objet de descripteur d'appareil de type 14 est expliquée au Tableau 3.

Tableau 3 – Définition de l'objet de descripteur d'appareil de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet de descripteur d'appareil de type 14 dans la MOB
2	Réservé	Lecture seule	Unsigned8	2	1	réservé
3	Type d'application	Lecture seule	Unsigned8	3	1	type d'application
4	Device ID	Lecture seule	VisibleString	4	32	identificateur de l'appareil
5	PD_Tag	Lecture seule	VisibleString	36	32	étiquette d'appareil
6	Active IP Address	Lecture seule	Unsigned32	68	4	adresse IP d'exploitation actuelle
7	Device Type	Lecture seule	Unsigned8	72	1	type d'appareil
8	Etat	Lecture seule	Unsigned8	73	1	état de l'appareil
9	Device Version	Lecture seule	Unsigned16	74	2	numéro de version de l'appareil
10	Annunciation Interval	Lecture seule	Unsigned16	76	2	intervalle auquel les appareils diffusent leur annonce
11	Annunciation Version Number	Lecture seule	Unsigned16	78	2	numéro de version de l'annonce de diffusion des appareils
12	Device Redundancy State	Lecture seule	Unsigned8	80	1	état de redondance de l'appareil
13	Device Redundancy Number	Lecture seule	Unsigned8	81	1	nombre de redondances de l'appareil
14	LANRedundancyPort	Lecture seule	Unsigned16	82	2	port de traitement des messages redondants de l'appareil
15	Max Redundancy Number	Lecture seule	Unsigned8	84	1	nombre maximum de redondances de l'appareil
16	Duplicate Tag Detected	Lecture seule	Booléen	85	1	propriété indiquant si l'étiquette (paramètre PD_Tag) de l'appareil est en conflit avec celle d'un autre appareil

4.2.3 Objet de synchronisation temporelle

La fonction de la classe d'objets de synchronisation temporelle est expliquée au Tableau 4:

Tableau 4 – Définition de l'objet de synchronisation temporelle

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet de synchronisation temporelle dans la MOB
2	Réservé	Lecture seule	OctetString	2	2	réservé
3	Primary Time Server	Lecture/Ecriture	Unsigned32	4	4	Adresse IP du serveur de synchronisation maître
4	Secondary Time Server	Lecture/Ecriture	Unsigned32	8	4	Adresse IP du serveur de synchronisation esclave
5	Time Request Timeout	Lecture seule	Unsigned32	12	4	délai maximum imparti au client de synchronisation pour obtenir la réponse du serveur de synchronisation (en secondes)
6	Time Request Interval	Lecture/Ecriture	Unsigned32	16	4	délai imparti au client de synchronisation pour envoyer une demande au serveur de synchronisation
7	Capable Time Sync Class	Lecture seule	Unsigned32	20	4	précision de synchronisation supportée par le client de synchronisation
8	Target Time Sync Class	Lecture/Ecriture	Unsigned32	24	4	précision de synchronisation exigée par le client de synchronisation
9	Current Time	Lecture seule	BinaryDate	28	8	heure de l'appareil
10	Standard Time Difference	Lecture seule	PrecisionTimeDifference	36	8	Standard time difference

4.2.4 Objet de temps de réponse maximum

La fonction de la classe d'objets de temps de réponse maximum est expliquée au Tableau 5.

Tableau 5 – Définition de l'objet de temps de réponse maximum

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet dans la MOB
2	Réservé	Lecture seule	OctetString	2	2	réservé
3	Max Response Time	Lecture/Ecriture	PrecisionTimeDifference	4	8	temps de réponse maximum du service confirmé en nanosecondes

4.2.5 Objet de gestion de planification des communications de type 14

La fonction de la classe d'objets de gestion de planification des communications de type 14 est expliquée au Tableau 6.

Tableau 6 – Définition de l'objet de gestion de la planification des communications de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet dans la MOB
2	Réservé	Lecture seule	OctetString	2	2	réservé
3	Communication MacroCycle	Lecture/Ecriture	PrecisionTimeDifference	4	8	macrocycle de communication du sous-réseau auquel appartient l'appareil. Unité: nanosecondes
4	NonPeriodic Data Transfer Offset	Lecture/Ecriture	PrecisionTimeDifference	12	8	laps de temps écoulé entre le début de la transmission d'un message non périodique et le début d'un macrocycle de communication. Unité: nanosecondes
5	Numéro de version du macrocycle de communication	Lecture seule	Unsigned16	20	2	numéro de version du macrocycle de communication

4.2.6 Objet donnant des informations sur l'application de l'appareil

La fonction de la classe d'informations sur l'application de l'appareil est expliquée au Tableau 7.

Tableau 7 – Définition de l'objet d'informations sur l'application de l'appareil

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet dans la MOB
2	XDDL Version	Lecture seule	Unsigned16	2	2	numéro de version de la description de l'appareil

4.2.7 En-tête d'informations sur l'application du bloc de fonctions

La fonction de la classe d'en-têtes d'informations sur l'application du bloc de fonctions est expliquée au Tableau 8.

Tableau 8 – Définition de l'en-tête d'informations sur l'application du bloc de fonctions

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet dans la MOB
2	Numéro de l'objet d'information d'application FB	Lecture seule	Unsigned16	2	2	nombre d'objets d'informations sur l'application du bloc de fonctions
3	Premier numéro de l'objet d'information d'application FB	Lecture seule	Unsigned16	4	2	premier numéro d'objet d'informations sur l'application du bloc de fonctions

4.2.8 En-tête d'informations sur l'application du domaine

La fonction de la classe d'en-têtes d'informations sur l'application du domaine est expliquée au Tableau 9.

Tableau 9 – Définition de l'en-tête d'informations sur l'application du domaine

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet d'en-tête d'informations sur l'application du domaine dans la MOB
2	Numéro de l'objet d'information d'application de domaine	Lecture seule	Unsigned16	2	2	nombre d'objets d'informations sur l'application du domaine dans l'appareil
3	Premier numéro de l'objet d'information de domaine	Lecture seule	Unsigned16	4	2	premier numéro d'objet d'informations sur l'application du domaine dans la MOB
4	Nombre d'objets de domaine configurés	Lecture seule	Unsigned16	6	2	nombre d'objets de domaine configurés
5	Nombre d'objets de domaine non configurés	Lecture seule	Unsigned16	8	2	nombre d'objets de domaine non configurés

4.2.9 En-tête d'objet de liaison de type 14

La fonction de la classe d'en-têtes d'objet de liaison FRT de type 14 est expliquée au Tableau 10.

Tableau 10 – Définition de l'en-tête d'objet de liaison de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'en-tête d'objet de liaison dans la MOB
2	Numéro de l'objet de liaison	Lecture seule	Unsigned16	2	2	numéro de l'objet de liaison dans l'appareil
3	Premier numéro de l'objet de liaison	Lecture seule	Unsigned16	4	2	premier numéro d'objet de liaison dans la MOB
4	Numéro de l'objet de liaison configuré	Lecture seule	Unsigned16	6	2	nombre d'objets de liaison configurés
5	Numéro de l'objet de liaison non configuré	Lecture seule	Unsigned16	8	2	nombre d'objets de liaison non configurés

4.2.10 En-tête d'objet de liaison de type 14

La fonction de la classe d'en-têtes d'objet de liaison FRT de type 14 est expliquée au Tableau 11.

Tableau 11 – Définition de l'en-tête d'objet de liaison FRT de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'en-tête d'objet de liaison FRT dans la MOB
2	Numéro de l'objet de liaison FRT	Lecture seule	Unsigned16	2	2	numéro de l'objet de liaison FRT dans l'appareil
3	Premier numéro de l'objet de liaison FRT	Lecture seule	Unsigned16	4	2	premier numéro d'objet de liaison FRT dans la MOB
4	Numéro de l'objet de liaison FRT configuré	Lecture seule	Unsigned16	6	2	nombre d'objets de liaison FRT configurés
5	Numéro de l'objet de liaison FRT non configuré	Lecture seule	Unsigned16	8	2	nombre d'objets de liaison FRT non configurés

4.2.11 Objet d'informations sur l'application du bloc de fonctions

La fonction de la classe d'informations sur l'application du bloc de fonctions est expliquée au Tableau 12.

Tableau 12 – Définition de l'objet d'informations sur l'application du bloc de fonctions

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet d'informations sur l'application du bloc de fonctions dans la MOB
2	Réservé	Lecture seule	Unsigned16	2	2	réservé
3	FB Name	Lecture seule	VisibleString	4	32	Nom du bloc de fonctions; sa longueur est 32 octets; si la longueur est inférieure à 32 octets, la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète
4	FB Type	Lecture seule	Unsigned16	36	2	FB Type
5	Numéro max d'instanciation	Lecture seule	Unsigned16	38	2	nombre maximum d'instances du bloc de fonctions
6	FB Execution Time	Lecture seule	Unsigned32	40	4	temps d'exécution du bloc de fonctions (en millisecondes)
7	First Number of Instantiation	Lecture seule	Unsigned16	44	2	premier numéro d'instance alloué à l'instance du bloc de fonctions lors de son instanciation

4.2.12 Objet de liaison de type 14

La fonction de la classe d'objets de liaison de type 14 est expliquée au Tableau 13.

Tableau 13 – Définition de l'objet de liaison de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet de liaison de type 14 dans la MOB
2	LocalAppID	Lecture/Ecriture	Unsigned16	2	2	identificateur de l'instance locale
3	Local Object ID	Lecture/Ecriture	Unsigned16	4	2	index de l'objet de variable local
4	RemoteAppID	Lecture/Ecriture	Unsigned16	6	2	identificateur du bloc de fonctions distant
5	RemoteObjectID	Lecture/Ecriture	Unsigned16	8	2	Identificateur de l'objet d'élément distant
6	ServiceOperation	Lecture/Ecriture	Unsigned8	10	1	Identificateur du service de type 14 utilisé par l'objet de liaison
7	ServiceRole	Lecture/Ecriture	Unsigned8	11	1	rôle de l'objet local dans le processus de communication
8	RemoteIPAddress	Lecture/Ecriture	Unsigned32	12	4	Adresse IP de l'appareil distant; si les objets des instances locale et de destination du bloc de fonctions se trouvent dans le même appareil de type 14, cette propriété peut être ignorée; si le service de type 14 utilise la méthode de diffusion

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
						ou de multidiffusion, il convient que cette propriété soit définie sur l'adresse du groupe de diffusion ou de multidiffusion
9	SendTimeOffset	Lecture/Ecriture	PrecisionTimeDifference	16	8	laps de temps écoulé entre le début de transmission d'un paquet périodique et le début du macro-cycle de communication. Son type de données est TimeDifference sur 4 octets. Unité: nanosecondes

4.2.13 Objet de liaison de type 14 FRT

La fonction de la classe d'objets de liaison FRT de type 14 est expliquée au Tableau 14.

Tableau 14 – Définition de l'objet de liaison FRT de type 14

N°	Nom de paramètre	Propriété de lecture/ écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet de liaison de type 14 dans la MOB
2	Local Object ID	Lecture/Ecriture	Unsigned16	2	2	index de l'objet de variable local
3	RemoteObjectID	Lecture/Ecriture	Unsigned16	4	2	Identificateur de l'objet d'élément distant
4	ServiceOperation	Lecture/Ecriture	Unsigned8	6	1	Identificateur du service de type 14 utilisé par l'objet de liaison
5	ServiceRole	Lecture/Ecriture	Unsigned8	7	1	rôle de l'objet local dans le processus de communication
6	RemoteMACAddress	Lecture/Ecriture	Unsigned32	8	4	Adresse MAC de l'appareil distant; si les objets des instances locale et de destination du bloc de fonctions se trouvent dans le même appareil de type 14, cette propriété peut être ignorée; si le service de type 14 utilise la méthode de diffusion ou de multidiffusion, il convient que cette propriété soit définie sur l'adresse du groupe de diffusion ou de multidiffusion
7	SendTimeOffset	Lecture/Ecriture	PrecisionTimeDifference	12	8	laps de temps écoulé entre le début de transmission d'un paquet périodique et le début du macro-cycle de communication. Son type de données est TimeDifference sur 4 octets. Unité: nanosecondes
8	ValidBitOffset	Lecture/Ecriture	Unsigned16	20	4	position relative du bit du moment où il convient que le message soit envoyé ou reçu par rapport à l'heure de début indiquée dans le champ DATA du paramètre de service FRTVariableDistribute
9	ValidBitNumber	Lecture/Ecriture	Unsigned16	24	4	numéro du bit du moment où il convient que le message concerné soit envoyé ou reçu par rapport à l'heure de début indiquée dans le champ DATA du paramètre de service FRTVariableDistribute

4.2.14 Objet d'informations sur l'application du domaine

La fonction de la classe d'informations sur l'application du domaine est expliquée au Tableau 15.

Tableau 15 – Définition de l'objet d'informations sur l'application du domaine

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets	Description
1	Object ID	Lecture seule	Unsigned16	0	2	index de l'objet d'informations sur l'application du domaine dans la MOB
2	ID d'objet de domaine	Lecture seule	Unsigned16	2	2	index de l'objet de domaine correspondant à l'objet d'informations sur l'application du domaine
3	ConfigurationStatus	Lecture seule	Booléen	4	1	état de configuration de l'objet de domaine. Paramètre de type booléen; si la valeur est true, cela signifie que l'objet de domaine n'est pas configuré
4	Réservé	Lecture seule	OctetString	5	3	réservé
5	Domain Name	Lecture seule	Unsigned16	8	32	nom de l'objet de domaine; si la longueur est inférieure à 32 octets, la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète

4.3 Définition des objets utilisés dans l'entité AAE de type 14

Le Paragraphe 4.3 définit les règles de codage des objets utilisés dans l'entité d'accès à l'application de type 14.

4.3.1 Objet de domaine

La fonction de la classe de domaines est expliquée au Tableau 16.

Tableau 16 – Définition de l'objet de domaine

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
1	ObjectID	Lecture seule	Unsigned16	2	index de l'objet de domaine
2	Domain Name	Lecture/Ecriture	VisibleString	32	nom de l'objet de domaine
3	Max octets	Lecture seule	Unsigned16	2	nombre maximum d'octets dans le domaine
4	Password	Lecture/Ecriture	Unsigned16	2	mot de passe utilisé pour accéder à l'objet de domaine
5	AccessGroups	Lecture/Ecriture	Unsigned8	1	groupe d'accès de l'objet de

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
					domaine
6	AccessRights	Lecture/Ecriture	Unsigned8	1	droit d'accès de l'objet de domaine
7	Local Address	Lecture seule	Unsigned32	4	pointeur désignant l'objet de domaine spécifique. Si ce paramètre n'est pas utilisé, il convient de le définir sur la valeur 0xFFFF FFFF
8	Domain State	Lecture seule	Unsigned8	1	état de l'objet de domaine; il peut avoir les valeurs suivantes: 0: EXISTENT 1: DOWNLOADING 2: UPLOADING 3: READY 4: IN-USE
9	Last State	Lecture seule	Unsigned8	1	état de l'objet de domaine avant chargement/téléchargement; il peut avoir les valeurs suivantes: 0: EXISTENT 1: DOWNLOADING 2: UPLOADING 3: READY 4: IN-USE
10	Used Application Counter	Lecture seule	Unsigned16	2	nombre de programmes qui utilisent le domaine; si la valeur du compteur est supérieure à 0, cela signifie que le domaine est utilisé et il ne peut donc pas être écrasé par le service de téléchargement; le type de données est unsigned16

4.3.2 Objet de variable simple

La définition de l'objet de variable simple est expliquée au Tableau 17.

Tableau 17 – Définition de l'objet de variable simple

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
1	ObjectID	Lecture seule	Unsigned16	2	index de l'objet de variable dans la MOB
2	Type de données	Lecture seule	Unsigned8	1	type de données de l'objet de variable
3	Longueur	Lecture seule	Unsigned16	2	longueur de l'objet de variable en octets
4	Local Address	Lecture seule	Unsigned32	4	pointeur désignant l'objet de variable spécifique, qui peut être utilisé pour désigner l'objet de domaine en interne. Si ce paramètre n'est pas utilisé, il convient de le définir sur la valeur 0xFFFF FFFF
5	Password	Lecture/Ecriture	Unsigned16	2	mot de passe utilisé pour accéder à l'objet de variable

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
6	AccessGroups	Lecture/Ecriture	Unsigned8	1	groupe d'accès de l'objet de variable
7	AccessRights	Lecture/Ecriture	Unsigned8	1	droit d'accès de l'objet de variable

4.3.3 Objet d'événement

La définition de l'objet d'événement est expliquée au Tableau 18.

Tableau 18 – Définition de l'objet d'événement

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
1	ObjectID	Lecture seule	Unsigned16	2	identificateur de l'objet d'événement
2	Longueur	Lecture seule	Unsigned16	2	longueur de l'objet d'événement en octets
3	Password	Lecture/Ecriture	Unsigned16	2	mot de passe utilisé pour accéder à l'objet de domaine
4	AccessGroups	Lecture/Ecriture	Unsigned8	1	groupe d'accès de l'objet de domaine
5	AccessRights	Lecture/Ecriture	Unsigned8	1	droit d'accès de l'objet de domaine
6	Local Address	Lecture seule	Unsigned32	4	pointeur désignant l'objet d'événement spécifique. Il est généralement utilisé pour désigner l'objet de variable en interne. Si ce paramètre n'est pas utilisé, il convient de le définir sur la valeur 0xFFFF FFFF
7	Enabled	Lecture seule	Booléen	1	Enabled = TRUE ⇔ UNLOCKED Signifie que l'objet d'événement est déverrouillé et que l'événement peut être envoyé Enabled = FALSE ⇔ LOCKED Signifie que l'objet d'événement est verrouillé et que l'événement ne peut pas être envoyé

4.3.4 Objet de mapping de ports de type 14

La définition de l'objet de mapping de ports de type 14 est expliquée au Tableau 19.

Tableau 19 – Définition de l'objet de mapping de ports de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
1	LocalIPAddress	Lecture seule	Unsigned32	4	Adresse IP de l'appareil local
2	RemoteIPAddress	Lecture seule	Unsigned32	4	Adresse IP de l'appareil distant
3	ActiveUdpPort	Lecture seule	Unsigned16	2	port UDP utilisé lors de l'envoi d'un message
4	ActiveServiceID	Lecture seule	Unsigned16	2	Service ID
5	ActiveMesssagLength	Lecture seule	Unsigned16	2	longueur du message en attente d'envoi
6	ActiveMessageID	Lecture seule	Unsigned16	2	Identificateur du paquet actif (ID message, par exemple)
7	ActiveMesssageTime	Lecture seule	Time Difference	6	temps de réponse maximum du

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
					message actif; si aucune réponse n'est requise, il convient que ce paramètre soit défini sur zéro
8	ActiveDataPointer	Lecture seule	Unsigned32	4	pointeur désignant l'en-tête du message actif
9	MaxMessageLength	Lecture seule	Unsigned16	2	longueur maximum autorisée d'un message. Si la longueur du message est supérieure à cette valeur, le système refusera l'envoi et émettra un indicateur d'erreur
10	MaxRetransmitNumber	Lecture seule	Unsigned16	2	nombre maximum de retransmissions autorisées

4.3.5 Objet de temporisateur de ports de type 14

La définition de l'objet de temporisateur de ports de type 14 est expliquée au Tableau 20.

Tableau 20 – Définition de l'objet de temporisateur de ports de type 14

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
1	TimerID	Lecture seule	Unsigned16	2	identificateur du temporisateur
2	ActiveServiceID	Lecture seule	Unsigned16	2	Identificateur du service utilisé pour envoyer le message
3	ActiveMessageID	Lecture seule	Unsigned16	2	identificateur du message actif, par exemple ID message indiqué dans le message
4	ActiveMessageTime	Lecture seule	Unsigned32	4	temps de réponse maximum du message actif; si aucune réponse n'est requise, ce paramètre doit être défini sur zéro. Unité: milliseconde. L'unité est en millisecondes pour les applications RT et en microsecondes pour les applications FRT

4.3.6 Objet ErrorType

La définition de l'objet ErrorType est expliquée au Tableau 21.

Tableau 21 – Définition de l'objet ErrorType

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
1	Error Class	Lecture seule	Int8	1	Classe d'erreurs
2	Error Code	Lecture seule	Int8	1	Code d'erreur, ce paramètre fournit une description plus détaillée de l'erreur
3	Additional Code	Lecture seule	Int8	1	Code supplémentaire, ce paramètre est facultatif; l'utilisateur peut définir son usage en fonction de ses besoins

N°	Nom de paramètre	Propriété de lecture/écriture	Type de données	Position relative en octets	Longueur en octets
4	Réservé	Lecture seule	Octetstring	1	réservé
5	Additional Description	Lecture seule	VisibleString	32	Description supplémentaire. Ce paramètre est facultatif; l'utilisateur peut s'en servir pour ajouter un texte descriptif de l'erreur. Son type de données est VisibleString

5 Syntaxe de transfert

5.1 Encodage des types de données de base

5.1.1 Booléen

Une valeur booléenne est codée sur un octet. Tous les bits sont définis sur 0 (FALSE) ou 1 (TRUE).

Si la valeur est TRUE, la valeur de chaque bit est indiquée dans le Tableau 22 (le bit 7 correspond au bit de poids fort et le bit 0 correspond au bit de poids faible).

Tableau 22 – Encodage de la valeur booléenne TRUE

Octet	Bit							
	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1

Si la valeur est FALSE, la valeur de chaque bit est indiquée dans le Tableau 23.

Tableau 23 – Encodage de la valeur booléenne FALSE

Octet	Bit							
	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0

5.1.2 Unsigned8

Le type Unsigned8 est codé sur un octet. La plage de valeurs s'étend de 0 à 255. Le poids de chaque bit est indiqué dans le Tableau 24.

Tableau 24 – Encodage du type de données Unsigned8

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

5.1.3 Unsigned16

Le type Unsigned16 est codé sur deux octets. La plage de valeurs s'étend de 0 à 216-1. Le poids de chaque bit est indiqué dans le Tableau 25.

Tableau 25 – Encodage du type de données Unsigned16

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

5.1.4 Unsigned32

Le type Unsigned32 est codé sur quatre octets. La plage de valeurs s'étend de 0 à 232-1. Le poids de chaque bit est indiqué dans le Tableau 26.

Tableau 26 – Encodage du type de données Unsigned32

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.5 Unsigned64

Le type Unsigned64 est codé sur quatre octets. La plage de valeurs s'étend de 0 à 264-1. Le poids de chaque bit est indiqué dans le Tableau 27.

Tableau 27 – Encodage du type de données Unsigned64

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.6 Int8

Le type Int8 est codé sur un octet; la plage s'étend de -128 à 127. Si le bit de signe (SN) est égal à 1, la valeur est négative; sinon, la valeur est positive; elle est nulle si le bit de signe est égal à 0. Le poids de chaque bit est indiqué dans le Tableau 28.

Tableau 28 – Encodage du type de données Int8

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.7 Int16

Le type Int16 est codé sur deux octets; la plage s'étend de -2^{15} à 2^{15}-1. Si le SN bit est égal à 1, la valeur est négative; sinon, la valeur est positive; elle est nulle si le bit de signe est égal à 0. Le poids de chaque bit est indiqué dans le Tableau 29.

Tableau 29 – Encodage du type de données Int16

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.8 Int32

Le type Int32 est codé sur quatre octets; la plage s'étend de -2^{31} à 2^{31}-1. Si le SN bit est égal à 1, la valeur est négative; sinon, la valeur est positive; elle est nulle si le bit de signe est égal à 0. Le poids de chaque bit est indiqué dans le Tableau 30.

Tableau 30 – Encodage du type de données Int32

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.9 Int64

Le type Int64 est codé sur huit octets; la plage s'étend de -2^{63} à $2^{63}-1$. Si la valeur du SN bit est égale à 1, la valeur est négative; sinon, la valeur est positive; elle est nulle si le bit de signe est égal à 0. Le poids de chaque bit est indiqué dans le Tableau 31.

Tableau 31 – Encodage du type de données Int64

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

5.1.10 Real

Le type Real est codé sur quatre octets. La plage de valeurs doit être conforme à la norme IEEE 754 relative aux nombres réels courts (32 bits au total). Si la valeur du SN bit est égale à 1, la valeur est négative; sinon, la valeur est positive; elle est nulle si le bit de signe est égal à 0. Les bits 6 à 0 de l'octet 1 et le bit 7 de l'octet 2 définissent le champ d'exposant. Il est suivi du champ des fractions qui va du bit 6 de l'octet 1 au bit 0 de l'octet 2. Le poids de chaque bit est indiqué dans le Tableau 32.

Tableau 32 – Encodage du type de données Real

Octet	Bit							
	7	6	5	4	3	2	1	0
1	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
2	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

5.1.11 VisibleString

Le type VisibleString est codé sous la forme d'une chaîne visible. La longueur est variable. Sa définition doit être conforme aux normes ISO/CEI 646 et ISO/CEI 2375. Le codage est représenté au Tableau 33.

Tableau 33 – Encodage du type de données VisibleString

Octet	Bit							
	7	6	5	4	3	2	1	0
1	premier caractère							
2	deuxième caractère							
……	etc.							
……	etc.							
N								

5.1.12 OctetString

Le type OctetString est codé sur un ou plusieurs octets qui sont alignés de 1 à n selon le numéro de séquence. Le codage est représenté au Tableau 34.

Tableau 34 – Encodage du type de données OctetString

Octet	Bit							
	7	6	5	4	3	2	1	0
1	Donnée binaire							
2	Donnée binaire							
……	……							
……	……							
N								

5.1.13 BitString

Le type BitString est codé dans un groupe d'octets. La longueur est variable et comprise entre 1 et n bits (où n représente un nombre naturel arbitraire). Le codage est représenté au Tableau 35.

Tableau 35 – Encodage du type de données BitString

Octet	Bit							
	7	6	5	4	3	2	1	0
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
……	etc.							
……	etc.							
n								

5.1.14 TimeOfDay

Le type TimeOfDay est codé sur 6 octets au total. Il se compose d'une date et d'une heure. Le champ Date est codé au format Unsigned16 (octets 1 et 2); il indique le nombre de jours écoulés depuis le premier jour de janvier 2000. Le premier jour de janvier 2000, la date commence à la valeur zéro. L'heure correspond au nombre de millisecondes écoulées depuis minuit; la valeur est réinitialisée à minuit. Le champ Time est codé au format Unsigned32 (octets 3 à 6). Le codage est représenté au Tableau 36.

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

Tableau 36 – Encodage du type de données TimeOfDay

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
3	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
4	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.15 BinaryDate

Le type BinaryDate est codé sur 8 octets; il se compose d'une date et d'une heure calendaires; l'encodage est représenté au Tableau 37.

Le champ Year est codé au format Unsigned16 (octet 1 et octet 2); si la valeur est égale à 2004, elle représente l'année 2004.

Le champ Month est codé au format Unsigned8 (octet 3). Il indique un des 12 mois de l'année. La plage de valeurs s'étend de 1 à 12.

Le champ Date est codé au format Unsigned8 (octet 4). Il indique un des 31 jours du mois. La plage de valeurs s'étend de 1 à 31.

Le champ Hour est codé au format Unsigned8 (octet 5). Il indique l'heure du jour, la plage de valeurs s'étend de 0 à 23 et sa valeur représente de 1 h à 24 h.

Le champ Minute est codé au format Unsigned8 (octet 6). Il indique le nombre de minutes, de 1 min à 60 min, écoulées dans l'heure. La plage de valeurs s'étend de 0 à 59.

Le champ Millisecond est codé au format Unsigned16 (octets 7 et 8). Il indique le nombre de millisecondes écoulées dans la minute. La plage de valeurs s'étend de 0 à 59999. Sa valeur représente 1 ms sur 60 000 ms dans une minute.

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

Tableau 37 – Encodage du type de données BinaryDate

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
5	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.16 PrecisionTimeDifference

Le type PrecisionTimeDifference est codé sur 8 octets; il indique un écart horaire et se compose d'un nombre de secondes et d'un nombre de nanosecondes.

Le champ Second est codé au format Unsigned32 (octets 1 à 4). Il indique le nombre de secondes de différence.

Le champ Nanosecond est codé au format Int32 (octets 5 à 8). Il indique le nombre de nanosecondes de différence. Le data type inclut également le symbole ns (pour la nanoseconde). Le codage est représenté au Tableau 38.

```
{
    Unsigned32 Second;        //second difference
    Int32 Nanosecond;         // Nanosecond with sign
}
```

Tableau 38 – Encodage du type de données PrecisionTimeDifference

Octet	Bit							
	7	6	5	4	3	2	1	0
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
5	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

5.1.17 Encodage de SEQUENCES

La structure SEQUENCE est comparable à un enregistrement. Elle représente un ensemble de données d'utilisateur de même type ou de type différent.

Une structure peut comporter une variable simple ou d'autres structures (définissant des composants).

5.1.18 Encodage de CHOIX

Un CHOICE (CHOIX) représente une sélection que l'utilisateur effectue parmi un ensemble de possibilités prédéfinies.

5.2 Encodage de l'en-tête de l'unité APDU de type 14

L'encodage de l'en-tête des paquets de messages de service de la couche application de type 14 est représenté au Tableau 39.

Tableau 39 – Encodage de l'en-tête de message de service de la couche application de type 14

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	ServiceID	Unsigned8	0	1	ce paramètre décrit le type de service ainsi que le type de message. Les bits 7 à 6 représentent le type de message: 00: message de requête 01: message de réponse 10: message d'erreur 11: réservé Les six bits inférieurs représentent l'identificateur du service
2	Réservé	OctetString	1	3	réservé
3	Longueur	Unsigned16	4	2	longueur du message entier
4	MessageID	Unsigned16	6	2	identificateur du message

5.3 Encodage des paramètres de service de l'entité de gestion de la couche FAL

5.3.1 Service EM_DetectingDevice

L'encodage des paramètres de demande du service EM_DetectingDevice est représenté au Tableau 40.

Tableau 40 – Encodage des paramètres de demande du service EM_DetectingDevice

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Query Type	Unsigned8	0	1	type de requête: 0: Requête portant sur une étiquette d'appareil physique. Le paramètre suivant est PD_Tag 1: Requête portant sur une étiquette de bloc de fonctions. Le paramètre suivant est FB Tag 2: Requête portant sur un élément. Les paramètres suivants sont FB Tag et ElementID
2	Réservé	Octetstring	1	3	réservé
3	PD_Tag	VisibleString	4	32	étiquette d'un appareil physique; si la longueur est inférieure à 32 octets, la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète
4	FB Tag	VisibleString	36	32	étiquette de l'instance du bloc de fonctions. Ce paramètre permet d'obtenir les informations relatives à un appareil de type 14 incluant l'instance du bloc de fonctions
5	Element ID	Unsigned16	68	2	Identificateur d'un élément figurant dans le bloc de fonctions. Le paramètre ElementID doit être utilisé conjointement avec le paramètre FB Tag, car il est spécifique à une seule instance du bloc de fonctions

5.3.2 Service EM_OnlineReply

L'encodage des paramètres de demande du service EM_OnlineReply est représenté au Tableau 41.

Tableau 41 – Encodage des paramètres de demande du service EM_OnlineReply

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Query Type	Unsigned8	0	1	type de requête: 0: Requête conformément à PD_Tag 1: Requête conformément à FB Tag 2: Requête conformément à ElementID
2	Duplicate Tag Detected	Booléen	1	1	propriété indiquant si l'étiquette (paramètre PD_Tag) de l'appareil est en conflit avec celle d'un autre ou d'autres appareils (valeurs PD_Tag en double, par exemple). TRUE= PD_Tags Collide
3	Réservé	Octetstring	2	2	réservé
4	Queried Object IP Address	Unsigned32	4	4	Adresse IP de l'appareil physique de type 14 (adresse IP de l'appareil local, par exemple)
5	Queried Object Device ID	VisibleString	8	32	identificateur de l'appareil physique de type 14 interrogé (identificateur de l'appareil local, par exemple). Si la longueur est inférieure à 32 octets, la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète
6	Queried Object PD_Tag	VisibleString	40	32	étiquette de l'appareil physique de type 14 interrogé (étiquette de l'appareil local, par exemple). Si la longueur est inférieure à 32 octets, la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète

5.3.3 Service EM_GetDeviceAttribute

5.3.3.1 Primitive de demande

L'encodage des paramètres de demande du service EM_GetDeviceAttribute est représenté au Tableau 42.

Tableau 42 – Encodage des paramètres de demande du service EM_GetDeviceAttribute

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Destination IP Addres	Unsigned32	0	4	Adresse IP de l'appareil cible

5.3.3.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service EM_GetDeviceAttribute est représenté au Tableau 43.

Tableau 43 – Encodage des paramètres de réponse positive du service EM_GetDeviceAttribute

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Device ID	VisibleString	0	32	Identificateur de l'appareil local
2	PD_Tag	VisibleString	32	32	étiquette d'un appareil physique; si la longueur est inférieure à 32 octets, la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète
3	Etat	Unsigned8	64	1	état de l'appareil de type 14: 0: aucune adresse 1: non configuré 2: configuré, opérationnel
4	Device Type	Unsigned8	65	1	type de l'appareil local
5	Annunciation Interval	Unsigned16	66	2	intervalle auquel le message d'annonce est envoyé par l'appareil
6	Annunciation Version Number	Unsigned16	68	2	numéro de version du message d'annonce
7	Duplicate Tag Detected	Booléen	70	1	propriété indiquant si l'étiquette (paramètre PD_Tag) de l'appareil est en conflit avec celle d'un autre ou d'autres appareils (valeurs PD_Tag en double, par exemple). TRUE= PD_Tag in collision
8	Redundancy Number	Unsigned8	71	1	nombre de redondances de l'appareil local; si l'appareil est actif, ce paramètre est égal à zéro et n'est pas suivi d'autres paramètres
9	Device Redundancy State	Unsigned8	72	1	état de redondance de l'appareil local: 0: statut actif 1: état de sauvegarde Si le nombre de redondances est égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse
10	Max Redundancy Number	Unsigned8	73	1	nombre maximum de redondances de l'appareil; si le nombre de redondances est

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
					égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse
11	Réservé	Octetstring	74	2	réservé
12	Active IP Address	Unsigned32	76	4	Adresse IP de l'appareil actif (adresse IP locale s'il n'y a pas de redondance). Si le nombre de redondances est égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse

5.3.3.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service EM_GetDeviceAttribute est représenté au Tableau 44.

Tableau 44 – Encodage des paramètres de réponse négative du service EM_GetDeviceAttribute

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationIPAddress	Unsigned32	0	4	Adresse IP de l'appareil cible
2	Error Type	ErrorType	4	N	voir ErrorType

5.3.4 Service EM_ActiveNotification

L'encodage des paramètres de demande du service EM_ActiveNotification est représenté au Tableau 45.

Tableau 45 – Encodage des paramètres de demande du service EM_ActiveNotification

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Device ID	VisibleString	0	32	Identificateur de l'appareil local
2	PD_Tag	VisibleString	32	32	étiquette physique d'un appareil local; sa longueur est de 32 octets et la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète
3	Etat	Unsigned8	64	1	état de l'appareil de type 14: 0: aucune adresse; 1: non configuré; 2: configuré, opérationnel
4	Device Type	Unsigned8	65	1	type de l'appareil local
5	Annunciation version number	Unsigned16	66	2	numéro de version du message d'annonce
6	Device Redundancy Number	Unsigned8	68	1	nombre de redondances de l'appareil local; si l'appareil est actif, ce paramètre est égal à 0; sinon, le paramètre suivant est non valide
7	Device Redundancy State	Unsigned8	69	1	état de redondance de l'appareil local: 0: état actif 1: état de sauvegarde Si le nombre de redondances est égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse
8	LAN Redundancy Port	Unsigned16	70	2	port de traitement des messages de redondance LAN de l'appareil qui est à l'origine de la demande de service
9	Duplicate Tag Detected	Booléen	72	1	propriété indiquant si l'étiquette (paramètre PD_Tag) de l'appareil est en conflit avec celle d'un autre ou d'autres appareils (valeurs PD_Tag en double, par exemple). TRUE= PD_Tags Collide
10	Réservé	Octetstring	73	2	réservé
11	Max. Redundancy Number	Unsigned8	75	1	nombre maximum de redondances de l'appareil
12	Active IP Address	Unsigned32	76	4	Adresse IP de l'appareil actif (adresse IP locale s'il n'y a pas de redondance). Si le nombre de redondances est égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse

5.3.5 Service EM_ConfiguringDevice

5.3.5.1 Primitive de demande

L'encodage des paramètres de demande du service EM_ConfiguringDevice est représenté au Tableau 46.

Tableau 46 – Encodage des paramètres de demande du service EM_ConfiguringDevice

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationIPAddress	Unsigned32	0	4	Adresse IP de destination
2	Device ID	VisibleString	4	32	Identificateur de l'appareil local
3	PD_Tag	VisibleString	36	32	étiquette physique d'un appareil local; sa longueur est de 32 octets et la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète
4	Annunciation Interval	Unsigned16	68	2	intervalle auquel le message d'annonce est envoyé par l'appareil
5	Duplicate Tag Detected	Booléen	70	1	propriété indiquant si l'étiquette (paramètre PD_Tag) de l'appareil est en conflit avec celle d'un autre ou d'autres appareils (valeurs PD_Tag en double, par exemple). TRUE= PD_Tags Collide
6	Device Redundancy Number	Unsigned8	71	1	nombre de redondances de l'appareil local; si l'appareil est actif, ce paramètre est égal à 0 et n'est pas suivi d'autres paramètres
7	LAN Redundancy Port	Unsigned16	72	2	port de traitement des messages de redondance LAN de l'appareil qui est à l'origine de la demande de service
8	Device Redundancy State	Unsigned8	74	1	état de redondance de l'appareil local: 0: état actif 1: état de sauvegarde Si le nombre de redondances est égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse
9	Max Redundancy Number	Unsigned8	75	1	nombre maximum de redondances de l'appareil
10	Active IP Address	Unsigned32	76	4	Adresse IP de l'appareil actif (adresse IP locale s'il n'y a pas de redondance). Si le nombre de redondances est égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse

5.3.5.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service EM_ConfiguringDevice est représenté au Tableau 47.

Tableau 47 – Encodage des paramètres de réponse positive du service EM_ConfiguringDevice

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Adresse IP de destination	Unsigned32	0	4	Adresse IP de destination
2	Max Redundancy Number	Unsigned8	4	1	nombre maximum de redondances de l'appareil; si le nombre de redondances est égal à 0, ce paramètre n'apparaît pas dans la primitive de réponse

5.3.5.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service EM_ConfiguringDevice est représenté au Tableau 48.

Tableau 48 – Encodage des paramètres de réponse négative du service EM_ConfiguringDevice

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Destination IP Addres	Unsigned32	0	4	Adresse IP de l'appareil cible
2	Error Type	ErrorType	4	N	voir Type d'erreur

5.3.6 Service EM_SetDefaultValue

5.3.6.1 Primitive de demande

L'encodage des paramètres de demande du service EM_SetDefaultValue est représenté au Tableau 49.

Tableau 49 – Encodage des paramètres de demande du service EM_SetDefaultValue

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationIPAddress	Unsigned32	0	4	Adresse IP de destination
2	Device ID	VisibleString	4	32	Identificateur de l'appareil local
3	PD_Tag	VisibleString	36	32	étiquette physique d'un appareil local; sa longueur est de 32 octets et la chaîne BLANK(0x20) est ajoutée à la place de la partie incomplète

5.3.6.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service EM_SetDefaultValue est représenté au Tableau 50.

Tableau 50 – Encodage des paramètres de réponse positive du service EM_SetDefaultValue

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Adresse IP de destination	Unsigned32	0	4	Adresse IP de destination

5.3.6.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service EM_SetDefaultValue est représenté au Tableau 51.

Tableau 51 – Encodage du paquet de refus du service de réinitialisation des attributs de l'appareil

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	Adresse IP de destination	Unsigned32	0	4	Adresse IP de destination
2	Error Type	ErrorType	4	N	voir Type d'erreur

5.4 Encodage des services AAE

5.4.1 Service DomainDownload

5.4.1.1 Primitive de demande

L'encodage des paramètres de demande du service DomainDownload est représenté au Tableau 52.

Tableau 52 – Encodage des paramètres de demande du service DomainDownload

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceAppID	Unsigned16	0	2	identificateur d'application de l'appareil source
2	DestinationAppID	Unsigned16	2	2	identificateur d'application de l'appareil de destination
3	DestinationObjectID	Unsigned16	4	2	identificateur d'objet de domaine de l'appareil de destination
4	DataNumber	Unsigned16	6	2	délais de téléchargement
5	MoreFollows	Booléen	8	1	indicateur signalant si d'autres téléchargements suivent
6	Réservé	OctetSring	9	1	réservé
7	DataLength	Unsigned16	10	2	longueur des données. La plage de valeurs s'étend de 0 à 512
8	Load Data	OctetString	12	N	données de téléchargement du domaine

5.4.1.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service DomainDownload est représenté au Tableau 53.

Tableau 53 – Encodage du paquet de réponse du service DomainDownload

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	adresse IP de destination

5.4.1.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service DomainDownload est représenté au Tableau 54.

Tableau 54 – Encodage des paramètres de réponse négative du service DomainDownload

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	adresse IP de destination
2	Réservé	Octetstring	2	2	réservé
3	ErrorType	ErrorType	4	N	voir Type d'erreur

5.4.2 Service DomainUpload

5.4.2.1 Primitive de demande

L'encodage des paramètres de demande du service DomainUpload est représenté au Tableau 55.

Tableau 55 – Encodage des paramètres de demande du service DomainUpload

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceAppID	Unsigned16	0	2	identificateur d'application de l'appareil source
2	DestinationAppID	Unsigned16	2	2	identificateur d'application de l'appareil de destination
3	DestinationObjectID	Unsigned16	4	2	identificateur d'objet de domaine de l'appareil de destination
4	DataNumber	Unsigned16	6	2	délais de téléchargement

5.4.2.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service DomainUpload est représenté au Tableau 56.

Tableau 56 – Encodage des paramètres de réponse positive du service DomainUpload

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	DataLength	Unsigned16	2	2	longueur des données; la plage de valeurs s'étend de 0 à 512
3	MoreFollows	Booléen	4	1	indicateur signalant si d'autres téléchargements suivent
4	Réservé	Octetstring	5	3	réservé
5	Load Data	Octetstring	8	N	données de chargement du domaine

5.4.2.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service DomainUpload est représenté au Tableau 57.

Tableau 57 – Encodage des paramètres de réponse négative du service DomainUpload

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.3 Service EventRoport

L'encodage des paramètres de demande du service EventRoport est représenté au Tableau 58.

Tableau 58 – Encodage des paramètres de demande du service EventRoport

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	SourceAppID	Unsigned16	2	2	identificateur d'application de l'appareil source
3	SourceObjectID	Unsigned16	4	2	identificateur d'objet de domaine de l'appareil source
4	EventNumber	Unsigned16	6	2	numéro de l'événement
5	EventData	OctetString	8	N	données d'un événement spécifique

5.4.4 Service EventRoportAcknowledge

5.4.4.1 Primitive de demande

L'encodage des paramètres de demande du service EventRoportAcknowledge est représenté au Tableau 59.

Tableau 59 – Encodage des paramètres de demande du service EventRoportAcknowledge

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	DestinationObjectID	Unsigned16	2	2	identificateur d'objet de l'appareil de destination
3	EventNumber	Unsigned16	4	2	numéro d'événement

5.4.4.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service EventRoportAcknowledge est représenté au Tableau 60.

Tableau 60 – Encodage des paramètres de réponse positive du service EventRoportAcknowledge

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination

5.4.4.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service EventRoportAcknowledge est représenté au Tableau 61.

Tableau 61 – Encodage des paramètres de réponse négative du service EventRoportAcknowledge

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.5 Service ReportConditionChanging

5.4.5.1 Primitive de demande

L'encodage des paramètres de demande du service ReportConditionChanging est représenté au Tableau 62.

Tableau 62 – Encodage des paramètres de demande du service ReportConditionChanging

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	DestinationObjectID	Unsigned16	2	2	identificateur d'objet de l'appareil de destination

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
3	Enabled	Booléen	4	1	indicateur d'activation
4	Réservé	Octetstring	5	3	réservé

5.4.5.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service ReportConditionChanging est représenté au Tableau 63.

Tableau 63 – Encodage des paramètres de réponse positive du service ReportConditionChanging

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination

5.4.5.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service ReportConditionChanging est représenté au Tableau 64.

Tableau 64 – Encodage des paramètres de réponse négative du service ReportConditionChanging

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.6 Service Read

5.4.6.1 Primitive de demande

L'encodage des paramètres de demande du service Read est représenté au Tableau 65.

Tableau 65 – Encodage des paramètres de demande du service Read

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	DestinationObjectID	Unsigned16	2	2	identificateur d'objet de l'appareil de destination
3	SubIndex	Unsigned16	4	2	Sous-index de l'objet accédé

5.4.6.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service Read est représenté au Tableau 66.

Tableau 66 – Encodage des paramètres de réponse positive du service Read

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Données	Octetstring	4	N	données renvoyées

5.4.6.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service Read est représenté au Tableau 67.

Tableau 67 – Encodage des paramètres de réponse négative du service Read

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.7 Service Write

5.4.7.1 Primitive de demande

L'encodage des paramètres de demande du service Write est représenté au Tableau 68.

Tableau 68 – Encodage des paramètres de demande du service Write

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	DestinationObjectID	Unsigned16	2	2	identificateur d'objet de l'appareil de destination
3	SubIndex	Unsigned16	4	2	Sous-index de l'objet écrit
4	Réservé	Octetstring	6	2	réservé
5	Données	Octetstring	8	N	données écrites

5.4.7.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service Write est représenté au Tableau 69.

Tableau 69 – Encodage des paramètres de réponse positive du service Write

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination

5.4.7.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service Write est représenté au Tableau 70.

Tableau 70 – Encodage des paramètres de réponse négative du service Write

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.8 Service VariableDistribute

L'encodage des paramètres de demande du service VariableDistribute est représenté au Tableau 71.

Tableau 71 – Encodage des paramètres de demande du service VariableDistribute

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceAppID	Unsigned16	0	2	identificateur d'application de l'appareil source
2	SourceObjectID	Unsigned16	2	2	identificateur d'objet de l'appareil source
3	Données	Octetstring	4	N	Données du service VariableDistribute

5.4.9 Service FRTRead

5.4.9.1 Primitive de demande

L'encodage des paramètres de demande du service FRTRead est illustré dans le Tableau 72.

Tableau 72 – Encodage des paramètres de demande du service FRTRead

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationObjectID	Unsigned16	0	2	identificateur d'objet de l'appareil de destination
2	SubIndex	Unsigned16	2	2	Sous-index de l'objet accédé

5.4.9.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service FRTRead est représenté au Tableau 73.

Tableau 73 – Encodage des paramètres de réponse positive du service FRTRead

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationObjectID	Unsigned16	0	2	identificateur d'objet de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Données	Octetstring	4	N	données renvoyées

5.4.9.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service FRTRead est représenté au Tableau 74.

Tableau 74 – Encodage des paramètres de réponse négative du service FRTRead

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationObjectID	Unsigned16	0	2	identificateur d'objet de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.10 Service FRTWrite

5.4.10.1 Primitive de demande

L'encodage des paramètres de demande du service FRTWrite est représenté au Tableau 75.

Tableau 75 – Encodage des paramètres de demande du service FRTWrite

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationObjectID	Unsigned16	0	2	identificateur d'objet de l'appareil de destination
2	SubIndex	Unsigned16	2	2	Sous-index de l'objet écrit
3	Réservé	Octetstring	4	2	réservé
4	Données	Octetstring	6	N	données écrites

5.4.10.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service FRTWrite est représenté au Tableau 76.

Tableau 76 – Encodage des paramètres de réponse positive du service FRTWrite

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationObjectID	Unsigned16	0	2	identificateur d'objet de l'appareil de destination

5.4.10.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service FRTWrite est représenté au Tableau 77.

Tableau 77 – Encodage des paramètres de réponse négative du service FRTWrite

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationObjectID	Unsigned16	0	2	identificateur d'objet de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.11 Service FRTVariableDistribute

L'encodage des paramètres de demande du service FRTVariableDistribute est représenté au Tableau 78.

Tableau 78 – Encodage des paramètres de demande du service FRTVariableDistribute

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceObjectID	Unsigned16	0	2	identificateur d'objet de l'appareil source
2	Données	Octetstring	2	N	Données du service VariableDistribute

5.4.12 Service BlockTransmissionOpen

5.4.12.1 Primitive de demande

Le codage des paramètres de demande du service BlockTransmissionOpen est représenté au Tableau 79.

Tableau 79 – Encodage des paramètres de demande du service BlockTransmissionOpen

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceAppID	Unsigned16	0	2	identificateur d'application de l'appareil source
2	DestinationAppID	Unsigned16	2	2	identificateur d'application de l'appareil de destination
3	DestinationObjectID	Unsigned16	4	2	identificateur d'objet de l'appareil de destination
4	BlockType	Unsigned16	6	2	type des données de bloc 0: vidéo; 1: audio; Autres: Réservé
5	BlockConfigInfo	OctetString	8	N	informations de configuration de la transmission par blocs

5.4.12.2 Primitive de réponse positive

Le codage des paramètres de réponse positive du service BlockTransmissionOpen est représenté au Tableau 80.

Tableau 80 – Encodage des paramètres de réponse positive du service BlockTransmissionOpen

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination

5.4.12.3 Primitive de réponse négative

Le codage des paramètres de réponse négative du service BlockTransmissionOpen est représenté au Tableau 81.

Tableau 81 – Encodage des paramètres de réponse négative du service BlockTransmissionOpen

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.13 Service BlockTransmissionClose

5.4.13.1 Primitive de demande

L'encodage des paramètres de demande du service BlockTransmissionClose est représenté au Tableau 82.

Tableau 82 – Encodage des paramètres de demande du service BlockTransmissionClose

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceAppID	Unsigned16	0	2	identificateur d'application de l'appareil source
2	DestinationAppID	Unsigned16	2	2	identificateur d'application de l'appareil de destination
3	DestinationObjectID	Unsigned16	4	2	identificateur d'objet de l'appareil de destination
4	BlockType	Unsigned16	6	2	type des données de bloc 0: vidéo; 1: audio; Autres: Réservé

5.4.13.2 Primitive de réponse positive

L'encodage des paramètres de réponse positive du service BlockTransmissionClose est représenté au Tableau 83.

Tableau 83 – Encodage des paramètres de réponse positive du service BlockTransmissionClose

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination

5.4.13.3 Primitive de réponse négative

L'encodage des paramètres de réponse négative du service BlockTransmissionClose est représenté au Tableau 84.

Tableau 84 – Encodage des paramètres de réponse négative du service BlockTransmissionClose

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	DestinationAppID	Unsigned16	0	2	identificateur d'application de l'appareil de destination
2	Réservé	Octetstring	2	2	réservé
3	Error Type	ErrorType	4	N	voir Type d'erreur

5.4.14 Service BlockTransmit

L'encodage des paramètres de demande du service BlockTransmit est représenté au Tableau 85.

Tableau 85 – Encodage des paramètres de demande du service BlockTransmit

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceAppID	Unsigned16	0	2	identificateur d'application de l'appareil source
2	DestinationAppID	Unsigned16	2	2	identificateur d'application de l'appareil de destination
3	DestinationObjectID	Unsigned16	4	2	identificateur d'objet de l'appareil de destination
4	DataLength	Unsigned16	6	2	Longueur des données
5	BlockType	Unsigned16	8	2	Type des données de bloc
6	SequenceNumber	Unsigned16	10	2	Numéro de séquence
7	Horodatage	PrecisionTimeDifference	12	8	Horodatage
8	SendCount	Unsigned16	20	2	Nombre total de paquets envoyés
9	BlockData	Octetstring	22	N	Données de bloc

5.4.15 Service BlockTransmissionHeartbeat

Le codage des paramètres de demande du service BlockTransmissionHeartbeat est représenté au Tableau 86.

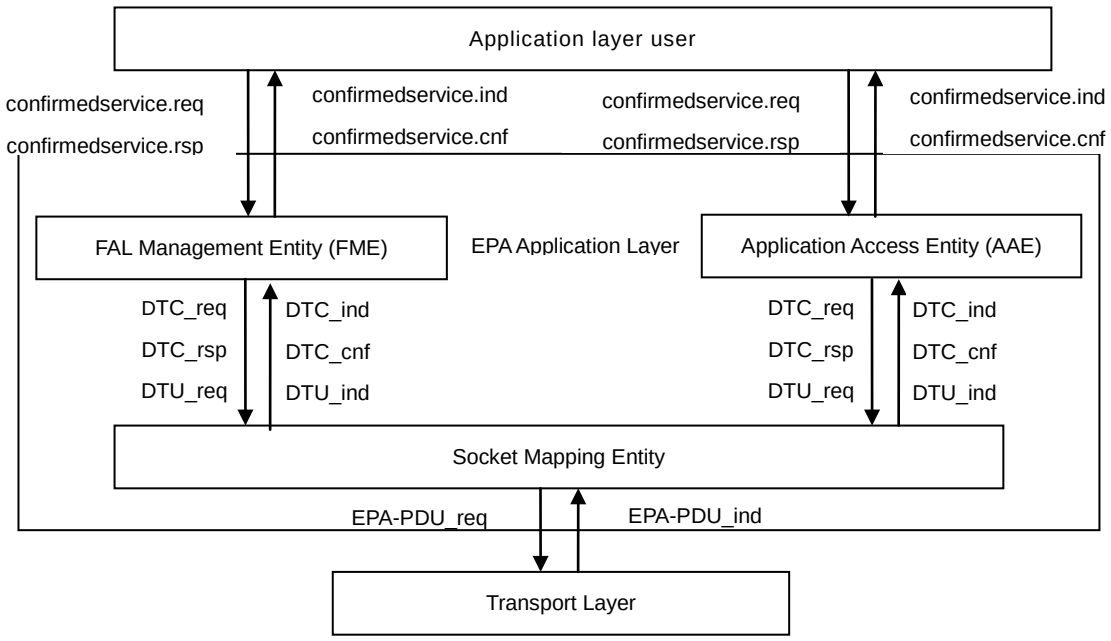
Tableau 86 – Encodage des paramètres de demande du service BlockTransmissionHeartbeat

N°	Nom de paramètre	Type de données	Position relative en octets	Longueur en octets	Description
1	SourceAppID	Unsigned16	0	2	identificateur d'application de l'appareil source
2	DestinationAppID	Unsigned16	2	2	identificateur d'application de l'appareil de destination
3	DestinationObjectID	Unsigned16	4	2	identificateur d'objet de l'appareil de destination
4	ReceptionCount	Unsigned16	6	2	nombre total de paquets reçus
5	CumulativeLost	Unsigned16	8	2	nombre total de paquets de données de bloc qui ont été perdus
6	Gigue	PrecisionTimeDifference	10	8	gigue de transmission calculée à partir du marqueur temporel

6 Structure des diagrammes d'états de protocole de la couche FAL

L'Article 6 décrit l'interface avec les services et les diagrammes protocolaires de couche FAL. Les conventions relatives aux descriptions sont indiquées dans la partie générale de la présente norme.

La Figure 2 illustre les relations existant entre les primitives. Les primitives échangées entre les différents éléments sont décrites ci-dessous.



Légende

Anglais	Français
Application layer user	Utilisateur de couche Application
FAL Management Entity (FME)	Entité FME

Anglais	Français
EPA Application Layer	Couche Application EPA
Application Access Entity (AAE)	Entité AAE
Socket Mapping Entity	Entité de mapping de ports
Transport Layer	Couche Transport

Figure 2 – Primitives échangées par le diagramme d'états protocolaire

7 Diagramme d'états de contexte AP

7.1 Primitives échangées entre ALU et ALE

Le Tableau 87 et le Tableau 88 présentent les primitives échangées entre l'utilisateur ALU et l'entité ALE.

Tableau 87 – Primitives remises par ALU à ALE

Nom de la primitive	Source	Paramètres associés	Fonctions
ConfirmedService.req	ALU	Data, Remote_IP_Address	Cette primitive permet à l'utilisateur ALU d'envoyer une primitive de demande de service confirmé à l'entité ALE
ConfirmedService.rsp	ALU	Data, Remote_IP_Address	Cette primitive permet à l'utilisateur ALU d'envoyer une primitive de réponse de service confirmé à l'entité ALE
UnconfirmedService.req	ALU	Data, Remote_IP_Address	Cette primitive permet à l'utilisateur ALU d'envoyer une primitive de demande de service non confirmé à l'entité ALE

Tableau 88 – Primitives remises par ALE à ALU

Nom de la primitive	Source	Paramètres associés	Fonctions
ConfirmedService.ind	ALE	Data, Remote_IP_Address	Cette primitive permet à l'entité ALE d'envoyer une primitive d'indication de service confirmé à l'utilisateur ALU
ConfirmedService.cnf	ALE	Data, Remote_IP_Address	Cette primitive permet à l'entité ALE d'envoyer une primitive de confirmation de service confirmé à l'utilisateur ALU
UnconfirmedService.ind	ALE	Data, Remote_IP_Address	Cette primitive permet à l'entité ALE d'envoyer une primitive d'indication de service non confirmé à l'utilisateur ALU

7.2 Descriptions du diagramme d'états de protocole

L'entité ACE de type 14 est toujours active. Son état est décrit dans le Tableau 89.

Tableau 89 – Descriptions des états de l'entité ACE

Etats	Descriptions
ACTIVE	Les entités ACE à l'état ACTIF sont prêtes à transférer des primitives de service à l'utilisateur ALU et l'entité FME (ou AAE), ou à recevoir des primitives provenant de l'utilisateur ALU et de l'entité FME (ou AAE)

7.3 Transitions d'états

Les transitions d'états de l'entité ACE sont présentées dans la Figure 3, le Tableau 90 et le Tableau 91.

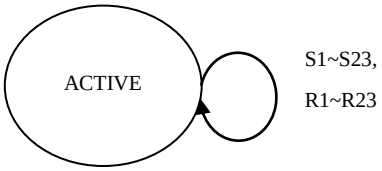


Figure 3 – Diagramme d'états de protocole de l'entité ACE

Tableau 90 – Transitions d'états de l'entité ACE (expéditeur)

#	Etat actuel	Evénement ou condition => action	Etat suivant

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

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

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

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

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

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

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

S8	ACTIVE	EM_SetDeviceAttribute.req =>	ACTIVE

#	Etat actuel	Evénement ou condition => action	Etat suivant

		confirmedservice.req {
		user_data:= Data,
		Destination_ip: = remote_ip_address,
		}	

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

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

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

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

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

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

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

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

S17	ACTIVE	EM_SetDeviceAttribute.rsp => confirmedservice.rsp {	ACTIVE

#	Etat actuel	Evénement ou condition => action	Etat suivant
		user_data:= Data, Destination_ip: = remote_ip_address, }	

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

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

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

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

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

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

Tableau 91 – Transitions d'états de l'entité ACE (destinataire)

#	Etat actuel	Evénement ou condition => action	Etat suivant

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

R2	ACTIVE	Confirmedservice.ind APServiceType(data) = "Domain Upload" =>	ACTIVE

#	Etat actuel	Evénement ou condition => action	Etat suivant

		DomainUpload.ind {
		Data:= user_data
		Destination_ip: = remote_ip_address,
		}	

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

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

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

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

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

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

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

#	Etat actuel	Evénement ou condition => action	Etat suivant

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

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

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

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

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

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

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

R17	ACTIVE	Confirmedservice.cnf APServiceType(data) = "EM_SetDeviceAttribute" => EM_SetDeviceAttribute.cnf { Data:= user_data	ACTIVE

#	Etat actuel	Evénement ou condition => action	Etat suivant
		Destination_ip: = remote_ip_address, }	

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

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

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

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

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

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

7.4 Descriptions des fonctions

Le Tableau 92 décrit les fonctions utilisées par les transitions d'états ACE.

Tableau 92 – Descriptions APServiceType()

Nom	APServiceType	Utilisée dans	ACE
Entrée		Sortie	
Données		Reçoit le type de service du message de service	
Fonction			
Cette fonction permet de déterminer le type du message de service reçu, notamment les primitives DomainDownload, DomainUpload, AcknowlodgeEventNotification, AlterEventConditionMonitor, Read, Write, EM_GetDeviceAttribute, EM_SetDeviceAttribute, EM_ClearDeviceAttribute, EventNotification, Distribute, EM_FindTagQuery, EM_FindTagReply et EM_DeviceAnnunciation			

8 Diagrammes d'états de l'entité FME

8.1 Primitives

8.1.1 Primitives échangées entre l'entité FME et l'utilisateur ALU

Le Tableau 93 et le Tableau 94 présentent les primitives échangées entre l'entité FME et l'utilisateur ALU.

Tableau 93 – Primitives remises par l'utilisateur ALU à l'entité FME

Nom de la primitive	Source	Paramètres associés	Fonctions
ConfirmedService.req	ALU	Data, Remote_IP_Address	Cette primitive permet à l'utilisateur ALU d'envoyer une primitive de demande de service confirmé à l'entité FME
ConfirmedService.rsp	ALU	Data, Remote_IP_Address	Cette primitive permet à l'utilisateur ALU d'envoyer une primitive de réponse de service confirmé à l'entité FME
UnconfirmedService.req	ALU	Data, Remote_IP_Address	Cette primitive permet à l'utilisateur ALU d'envoyer une primitive de demande de service non confirmé à l'entité FME

Tableau 94 – Primitives remises par l'entité FME à l'utilisateur ALU

Nom de la primitive	Source	Paramètres associés	Fonctions
ConfirmedService.ind	FME	Data, Remote_IP_Address	Cette primitive permet à l'entité FME d'envoyer une primitive d'indication de service confirmé à l'utilisateur ALU
ConfirmedService.cnf	FME	Data, Remote_IP_Address	Cette primitive permet à l'entité FME d'envoyer une primitive de confirmation de service confirmé à l'utilisateur ALU
UnconfirmedService.ind	FME	Data, Remote_IP_Address	Cette primitive permet à l'entité FME d'envoyer une primitive d'indication de service non confirmé à l'utilisateur ALU

8.1.2 Paramètres de primitives échangés entre l'entité FME et l'utilisateur ALU

Le Tableau 95 présente les paramètres de primitives échangés entre l'entité FME et l'utilisateur ALU.

Tableau 95 – Paramètres de primitives échangés entre l'entité FME et l'utilisateur ALU

Nom de paramètre	Description
Remote_ip_address	Ce paramètre transfère l'adresse IP de l'appareil distant, c'est-à-dire l'adresse de destination envoyée par l'expéditeur et l'adresse source reçue par le destinataire
Données	Ce paramètre transfère les données envoyées par l'expéditeur ainsi que les données reçues par le destinataire

8.1.3 Primitives échangées entre l'entité FME et l'entité ESME

Le Tableau 96 et le Tableau 97 présentent les primitives échangées entre l'entité FME et l'entité ESME.

Tableau 96 – Primitives remises par l'entité FME à l'entité ESME

Nom de la primitive	Source	Paramètres associés	Fonctions
DTC_req	FME	remote_ip_address, Données	Cette primitive permet à l'entité FME d'envoyer une primitive de demande de service confirmé à l'entité ESME
DTC_rsp	FME	remote_ip_address, Données	Cette primitive permet à l'entité FME d'envoyer une primitive de réponse de service confirmé à l'entité ESME
DTU_req	FME	remote_ip_address, Données	Cette primitive permet à l'entité FME d'envoyer une primitive de demande de service non confirmé à l'entité ESME

Tableau 97 – Primitives remises par l'entité ESME à l'entité FME

Nom de la primitive	Source	Paramètres associés	Fonctions
DTC_ind	Entité de mapping de ports de type 14	remote_ip_address, Données	Cette primitive permet à l'entité ESME d'envoyer une primitive d'indication de service confirmé à l'entité FME
DTC_cnf	Entité de mapping de ports de type 14	remote_ip_address, Données	Cette primitive permet à l'entité ESME d'envoyer une primitive d'acquittement de service confirmé à l'entité FME
DTU_ind	Entité de mapping de ports de type 14	remote_ip_address, Données	Cette primitive permet à l'entité ESME d'envoyer une primitive d'indication de service non confirmé à l'entité FME

8.1.4 Paramètres de primitives échangés entre l'entité FME et l'entité ESME

Le Tableau 98 présente les paramètres de primitives échangés entre l'entité FME et l'entité ESME.

Tableau 98 – Paramètres de primitives échangés entre l'entité FME et l'entité ESME

Nom de paramètre	Description
Remote_ip_address	Ce paramètre transfère l'adresse IP de l'appareil distant, c'est-à-dire l'adresse de destination envoyée par l'expéditeur et l'adresse source reçue par le destinataire
Données	Ce paramètre transfère les données envoyées par l'expéditeur ainsi que les données reçues par le destinataire

8.2 Descriptions du diagramme d'états de protocole

Les appareils de type 14 peuvent être dans l'état *Aucune adresse*, *Non configuré* ou *Configuré*. Le diagramme d'états de protocole correspondant est illustré dans la Figure 4.

8.2.1 Aucune adresse

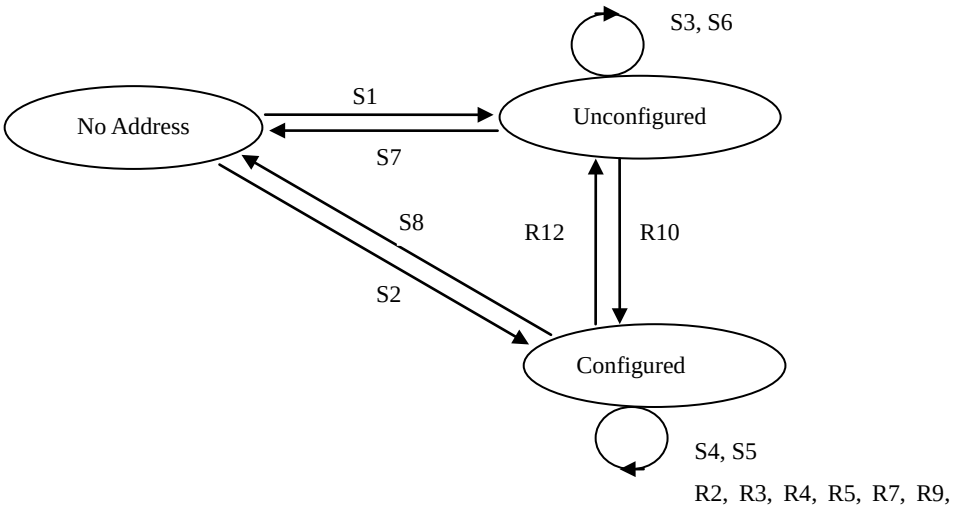
Dans cet état, l'appareil de type 14 attend l'affectation d'une adresse IP. L'adresse IP peut être affectée de manière statique par l'utilisateur ou de manière dynamique par le serveur DHCP. Une fois que l'appareil réceptionne son adresse IP par le biais du serveur DHCP, il passe dans l'état suivant (*Non configuré* ou *Configuré*) selon l'état dans lequel il se trouvait avant d'être mis hors tension. Autrement dit, si l'appareil était à l'état *Configuré* au moment de sa mise hors tension, il passera à l'état *Configuré* (et vice-versa).

8.2.2 Non configuré

Dans cet état, l'entité FME utilise l'adresse IP de destination de multidiffusion pour envoyer un message de demande EM_ActiveNotification afin d'informer les autres appareils de sa présence. Une fois le message de demande reçu par les appareils, l'application de configuration de l'utilisateur peut demander, effacer ou définir les attributs de l'appareil par le biais des services EM_DetectingDevice, EM_ConfiguringDevice et EM_SetDefaultValue. A l'issue de la configuration, l'appareil passe à l'état *Configuré* et s'exécute en mode de fonctionnement normal.

8.2.3 Configured

Dans cet état, l'appareil peut participer aux opérations de fonctionnement normal. Les utilisateurs peuvent configurer les informations sur son application au moyen des services fournis par la couche Application afin de mettre en œuvre une fonction de contrôle prédéfinie spécifique.



Légende

Anglais	Français
Configured	Configuré
Unconfigured	Non configuré
No Address	Sans adresse

Figure 4 – Diagramme d'états de protocole de l'entité FME

8.3 Transitions d'états

Les passages d'état de l'entité FME sont présentés dans la Figure 4 et le Tableau 99.

Tableau 99 – Transitions d'états de l'entité FME de type 14

#	Etat actuel	Evénement ou condition => action	Etat suivant
S1	Aucune adresse	RcvNewIpAddress (address) = TRUE && Attribute_Set ()= FALSE => RestoreDefaults () NewAddress(address) EM_ActiveNotification.req {} Restart_Type 14RepeatTimer()	Non configuré

#	Etat actuel	Evénement ou condition => action	Etat suivant
S2	Aucune adresse	RcvNewIpAddress (address) = TRUE && Attribute_Set()=TRUE => Clear_DuplicatePdTagFlag() NewAddress (address) EM_ActiveNotification.req {} Restart_Type 14RepeatTimer () EM_DetectingDevice.req {}	Configuré
S3	Non configuré	RepeatTimerExpires () => EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	Non configuré
S4	Configuré	EM_UnconfirmedService.req {} || EM_ConfirmedService.req {} || EM_ConfirmedService.rsp {} => Send_EM_ReqRspMessage (em_svc)	Configuré
S5	Configuré	EM_ConfirmedService.err {} => Send_EM_CommonErrorRsp ()	Configuré
S6	Non configuré	SntpSyncLost () => EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	Non configuré
S7	Non configuré	IPAddressCollision () = TRUE = > (aucune action entreprise)	Aucune adresse
S8	Configuré	IPAddressCollision () = TRUE = > (aucune action entreprise)	Aucune adresse
R1	Non configuré	RecvMsg () ="Any Confirmed Type 14_FME Rsp Message" || RecvMsg ()="Any Confirmed Type 14_FME Error Message" => EM_ConfirmedService.cnf{}	Non configuré
R2	Configuré	RecvMsg()=" EM_DetectingDevice" && QueryMatch (em_svc) = TRUE => EM_OnlineReply.req {}	Configuré
R3	Configuré	RecvMsg()=" EM_OnlineReply" && _MessageIdMatch(em_svc) = TRUE && DeviceId_Match (em_svc) = FALSE => Type 14Set_DuplicatePdTagFlag () EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	Configuré
R4	Configuré	RecvMsg()=" EM_OnlineReply" && _MessageIDMatch(em_svc) = TRUE && DeviceId_Match (em_svc) = TRUE => //Do nothing-the response is from this device	Configuré

#	Etat actuel	Evénement ou condition => action	Etat suivant
R5	Configuré	RecvMsg() = " EM_OnlineReply" => EM_OnlineReply.ind {}	Configuré
R6	Non configuré	RecvMsg() = " EM_GetDeviceAttributeReq" => EM_ConfrimedService.err {}	Non configuré
R7	Configuré	RecvMsg()=" EM_GetDeviceAttributeReq" => EM_GetDeviceAttribute.rsp {}	Configuré
R8	Non configuré	RecvMsg()=" EM_ConfiguringDeviceReq" || RecvMsg()=" EM_SetDefaultValueReq" && DeviceId_Match(em_svc)= FALSE => EM_ConfirmedService.err { }	Non configuré
R9	Configuré	RecvMsg()="EM_ConfiguringDeviceReq" || RecvMsg()=" EM_SetDefaultValueReq" && PdTag_Match(em_svc)= TRUE => EM_ConfirmedService.rsp {}	Configuré
R10	Non configuré	RecvMsg()=" EM_ConfiguringDeviceReq" => Set_Attribute_Data(em_svc) Clear_DuplicatePdTagFlag () EM_ConfiguringDevice.rsp {} EM_ActiveNotification.req {} Restart_Type 14RepeatTimer () EM_DetectingDevice.req {}	Configuré
R11	Configuré	RecvMsg=" EM_GetDeviceAttributeReq" => EM_GetDeviceAttribute.rsp {}	Configuré
R12	Configuré	RecvMsg()="EM_SetDefaultValueReq" => ResoreDefaults () EM_SetDefaultValue.rsp {} EM_ActiveNotification.req {} Restart_Type 14RepeatTimer ()	Non configuré

8.4 Descriptions des fonctions

Les fonctions utilisées par les passages d'état de l'entité FME sont décrites dans les tableaux qui suivent, du Tableau 100 au Tableau 117.

NOTE Le paramètre em_svc représente le message envoyé par les programmes de configuration de l'utilisateur.

8.4.1 RcvNewIpAddress()

La fonction RcvNewIpAddress() est décrite dans le Tableau 100.

Tableau 100 – Descriptions RcvNewIpAddress()

Nom	RcvNewIpAddress	Utilisation	FME
Entrée		Sortie	
Adresse		TRUE ou FALSE	
Fonction			
Cette fonction est appelée lorsqu'une adresse IP est reçue. Les paramètres d'entrée identifient l'interface de l'appareil ainsi que l'adresse IP reçue. Si cette fonction est appelée correctement, elle renvoie la valeur TRUE. Sinon, elle renvoie la valeur FALSE			

8.4.2 Attribute_Set()

La fonction Attribute_Set() est décrite dans le Tableau 101.

Tableau 101 – Descriptions Attribute_Set()

Nom	Attribute_Set	Utilisation	FME
Entrée		Sortie	
Aucun		TRUE ou FALSE	
Fonction			
Cette fonction renvoie la valeur TRUE si l'attribut de l'appareil de type 14 a été défini par le service EM_ConfiguringDevice et qu'il est toujours valide. Sinon, elle renvoie la valeur FALSE			

8.4.3 RestoreDefaults()

La fonction RestoreDefaults() est décrite dans le Tableau 102.

Tableau 102 – Descriptions RestoreDefaults()

Nom	RestoreDefaults	Utilisation	FME
Entrée		Sortie	
Aucun		Aucun	
Fonction			
Lorsque cette fonction est appelée, toutes les liaisons sont déconnectées et les attributs de l'élément EME sont définis sur la valeur par défaut			

8.4.4 NewAddress()

La fonction NewAddress() est décrite dans le Tableau 103.

Tableau 103 – Descriptions NewAddress()

Nom	NewAddress	Utilisation	FME
Entrée		Sortie	
Adresse		TRUE ou FALSE	
Fonction			
Cette fonction renvoie la valeur TRUE si l'adresse IP est mise à jour correctement lorsque l'appareil de type 14 reçoit une nouvelle adresse IP. Sinon, elle renvoie la valeur FALSE			

8.4.5 Restart_Type 14RepeatTimer()

La fonction Restart_Type 14RepeatTimer() est décrite dans le Tableau 104.

Tableau 104 – Descriptions Restart_Type 14RepeatTimer()

Nom	Restart_Type 14RepeatTimer	Utilisation	FME
Entrée		Sortie	
Aucun		Aucun	
Fonction			
Cette fonction est appelée pour restaurer et relancer le temporisateur de ports de type 14			

8.4.6 Clear_DuplicatePdTagFlag()

La fonction Clear_DuplicatePdTagFlag() est décrite dans le Tableau 105.

Tableau 105 – Descriptions Clear_DuplicatePdTagFlag()

Nom	Clear_DuplicatePdTagFlag	Utilisation	FME
Entrée		Sortie	
Aucun		Aucun	
Fonction			
Cette fonction est appelée pour effacer un état détecté en double.			

8.4.7 Type 14RepeatTimerExpire()

La fonction Type 14RepeatTimerExpire() est décrite dans le Tableau 106.

Tableau 106 – Descriptions Type 14RepeatTimerExpire()

Nom	Type 14RepeatTimerExpire	Utilisation	FME
Entrée		Sortie	
Aucun		Aucun	
Fonction			
Cette fonction est appelée lorsque le temporisateur de ports de type 14 arrive à expiration			

8.4.8 Send_EM_ReqRspMessage()

La fonction Send_EM_ReqRspMessage() est décrite dans le Tableau 107.

Tableau 107 – Descriptions Send_EM_ReqRspMessage()

Nom	Send_EM_ReqRspMessage	Utilisation	FME
Entrée		Sortie	
Em_svc		Aucun	
Fonction			
Cette fonction permet de créer et d'envoyer des messages de demande ou de réponse			

8.4.9 Send_EM_CommonErrorRsp()

La fonction Send_EM_CommonErrorRsp() est décrite dans le Tableau 108.

Tableau 108 – Descriptions Send_EM_CommonErrorRsp()

Nom	Send_EM_CommonErrorRsp	Utilisation	FME
Entrée		Sortie	
service_type, Spec_params		Aucun	
Fonction			
Cette fonction permet de créer et d'envoyer des informations de réponse d'erreur			

8.4.10 SntpSyncLost()

La fonction SntpSyncLost() est décrite dans le Tableau 109.

Tableau 109 – Descriptions SntpSyncLost()

Nom	SntpSyncLost	Utilisation	FME
Entrée		Sortie	
Aucun		Aucun	
Fonction			
Cette fonction est appelée lorsque l'état de synchronisation entre l'appareil et le serveur de synchronisation distant sélectionné est perdu			

8.4.11 IPAddressCollision()

La fonction IPAddressCollision() est décrite dans le Tableau 110.

Tableau 110 – Descriptions IPAddressCollision()

Nom	IPAddressCollision	Utilisation	FME
Entrée		Sortie	
Aucun		TRUE ou FALSE	
Fonction			
Cette fonction renvoie la valeur TRUE si l'adresse IP entre en conflit avec une autre. Sinon, elle renvoie la valeur FALSE			

8.4.12 RecvMsg()

La fonction RecvMsg() est décrite dans le Tableau 111.

Tableau 111 – Descriptions RecvMsg()

Nom	RecvMsg	Utilisation	FME
Entrée		Sortie	
Em_svc		Type de message Em_svc	
Fonction			
Cette fonction est appelée lorsqu'un message est reçu. Elle déchiffre le paramètre em_svc et renvoie le type du service de gestion de type 14			

8.4.13 QueryMatch()

La fonction QueryMatch() est décrite dans le Tableau 112.

Tableau 112 – Descriptions QueryMatch()

Nom	QueryMatch	Utilisation	FME
Entrée		Sortie	
Em_svc		TRUE ou FALSE	
Fonction			
Cette fonction renvoie la valeur TRUE si l'objet demandé se trouve dans l'appareil			

8.4.14 MessageIDMatch()

La fonction MessageIDMatch() est décrite dans le Tableau 113.

Tableau 113 – Descriptions MessageIDMatch()

Nom	MessageIDMatch	Utilisation	FME
Entrée		Sortie	
Em_svc		TRUE ou FALSE	
Fonction			
Cette fonction renvoie la valeur TRUE si le paramètre MessageID contenu dans les messages correspond à celui du service EM_DetectingDevice			

8.4.15 DeviceId_Match()

La fonction DeviceId_Match() est décrite dans le Tableau 114.

Tableau 114 – Descriptions DeviceId_Match()

Nom	DeviceId_Match	Utilisation	FME
Entrée		Sortie	
em_svc		TRUE ou FALSE	
Fonction			
Cette fonction renvoie la valeur TRUE si le paramètre DeviceID contenu dans le message correspond exactement à celui de l'appareil			

8.4.16 PdTag_Match()

La fonction PdTag_Match() est décrite dans le Tableau 115.

Tableau 115 – Descriptions PdTag_Match()

Nom	PdTag_Match	Utilisation	FME
Entrée		Sortie	
em_svc		TRUE ou FALSE	
Fonction			
Cette fonction renvoie la valeur TRUE si le paramètre PD_Tag contenu dans le message correspond à celui de l'appareil			

8.4.17 Set_Attribute_Data()

La fonction Set_Attribute_Data() est décrite dans le Tableau 116.

Tableau 116 – Descriptions Set_Attribute_Data()

Nom	Set_Attribute_Data	Utilisation	FME
Entrée		Sortie	
em_svc		Aucun	
Fonction			
Cette fonction permet de mettre à jour les attributs de l'élément EME dans l'appareil de type 14			

8.4.18 Set_DuplicatePdTagFlag()

La fonction Set_DuplicatePdTagFlag() est décrite dans le Tableau 117.

Tableau 117 – Descriptions Set_DuplicatePdTagFlag()

Nom	Set_DuplicatePdTagFlag	Utilisation	FME
Entrée		Sortie	
Aucun		Aucun	
Fonction			
La fonction DuplicateDetectedState est définie si l'appareil a détecté qu'un autre appareil possédait le même paramètre PD_Tag			

9 Diagramme d'états de protocole de l'entité AAE

9.1 Primitives

9.1.1 Primitives échangées entre l'entité AAE et l'utilisateur ALU

Le Tableau 118 et le Tableau 119 décrivent les primitives échangées entre l'entité AAE et l'utilisateur ALU.

Tableau 118 – Primitives adressées par l'utilisateur ALU à l'entité AAE

Nom de la primitive	Source	Paramètres associés	Fonctions
ConfirmedService.req	ALU	remote_ip_address, données	Cette primitive permet à l'utilisateur ALU de transmettre une primitive de demande ConfirmedService à l'entité AAE
ConfirmedService.rsp	ALU	remote_ip_address, données	Cette primitive permet à l'utilisateur ALU de transmettre une primitive de réponse ConfirmedService à l'entité AAE
UnconfirmedService.req	ALU	remote_ip_address, données	Cette primitive permet à l'utilisateur ALU de transmettre une primitive de demande UnconfirmedService à l'entité AAE

Tableau 119 – Primitives adressées par l'entité AAE à l'utilisateur ALU

Nom de la primitive	Source	Paramètres associés	Fonctions
ConfirmedService.ind	AAE	remote_ip_address, données	Cette primitive permet à l'entité AAE de transmettre une primitive d'indication ConfirmedService à l'utilisateur ALU
ConfirmedService.cnf	AAE	remote_ip_address, données	Cette primitive permet à l'entité AAE de transmettre une primitive de confirmation ConfirmedService à l'utilisateur ALU
UnconfirmedService.ind	AAE	remote_ip_address, données	Cette primitive permet à l'entité AAE de transmettre une primitive d'indication UnconfirmedService à l'utilisateur ALU

9.1.2 Paramètres de primitives échangés entre l'entité AAE et l'utilisateur ALU

Le Tableau 120 décrit les paramètres utilisés dans les primitives échangées entre l'entité AAE et l'utilisateur ALU.

Tableau 120 – Paramètres de primitives échangés entre l'entité AAE et l'utilisateur ALU

Nom de paramètre	Description
remote_ip_address	Ce paramètre transfère l'adresse IP de l'appareil distant, c'est-à-dire l'adresse de destination à laquelle l'expéditeur envoie des données et l'adresse source des données reçues par le destinataire
Données	Ce paramètre transfère les données envoyées par l'expéditeur ainsi que les données reçues par le destinataire

9.1.3 Primitives échangées entre l'entité AAE et l'entité ESME

Le Tableau 121 et le Tableau 122 décrivent les primitives échangées entre l'entité AAE et l'entité ESME.

Tableau 121 – Primitives adressées par l'entité AAE à l'entité ESME

Nom de la primitive	Source	Paramètres associés	Fonctions
DTC.req	AAE	remote_ip_address, données	Cette primitive permet à l'entité AAE de transmettre une primitive de demande ConfirmedService à l'entité ESME
DTC.rsp	AAE	remote_ip_address, données	Cette primitive permet à l'entité AAE de transmettre une primitive de réponse ConfirmedService à l'entité ESME
DTU.req	AAE	remote_ip_address, données	Cette primitive permet à l'entité AAE de transmettre une primitive de demande UnconfirmedService à l'entité ESME

Tableau 122 – Primitives adressées par l'entité ESME à l'entité AAE

Nom de la primitive	Source	Paramètres associés	Fonctions
DTC.ind	ESME	remote_ip_address, données	Cette primitive permet à l'entité ESME de transmettre une primitive d'indication ConfirmedService à l'entité AAE
DTC.cnf	ESME	remote_ip_address, données	Cette primitive permet à l'entité ESME de transmettre une primitive de confirmation ConfirmedService à l'entité AAE
DTU.ind	ESME	remote_ip_address, données	Cette primitive permet à l'entité ESME de transmettre une primitive d'indication UnconfirmedService à l'entité AAE

9.1.4 Paramètres de primitives échangés entre l'entité AAE et l'entité ESME

Le Tableau 123 décrit les paramètres utilisés dans les primitives échangées entre le point de fin AAE et l'entité ESME.

Tableau 123 – Paramètres de primitives échangés entre l'entité AAE et l'entité ESME

Nom de paramètre	Description
remote_ip_address	Ce paramètre transfère l'adresse IP de l'appareil distant, c'est-à-dire l'adresse de destination à laquelle l'expéditeur envoie des données et l'adresse source des données reçues par le destinataire
Données	Ce paramètre transfère les données envoyées par l'expéditeur ainsi que les données reçues par le destinataire

9.2 Diagramme d'états de l'entité AAE

9.2.1 Descriptions des états de l'entité AAE

L'entité AAE de type 14 est toujours active. Son état est décrit dans le Tableau 124.

Tableau 124 – Descriptions des états de l'entité AAE

Etats	Descriptions
ACTIVE	Les entités ACE à l'état ACTIVE sont prêtes à transférer des primitives de service à l'utilisateur ALU et à l'entité ESME, ou à recevoir des primitives de l'utilisateur ALU et de l'entité ESME

9.2.2 Transitions d'états

Le diagramme d'états protocolaire de l'entité AAE est illustré à la Figure 5 et les tableaux qui suivent de Tableau 125 à Tableau 126.

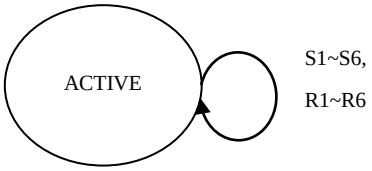


Figure 5 – Diagramme de transitions d'états de l'AAE

Tableau 125 – Transitions d'états de l'entité AAE (expéditeur)

#	Etat actuel	Evénement ou condition => action	Etat suivant

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

#	Etat actuel	Evénement ou condition => action	Etat suivant
		}	

S6	ACTIVE	UnconfirmedService.req
		=>
		DTU.req {
		user_data:= Data,
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

Tableau 126 – Transitions d'états de l'entité AAE (destinataire)

#	Etat actuel	Evénement ou condition => action	Etat suivant

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

9.2.3 Descriptions des fonctions

Les fonctions utilisées par les passages d'état de l'entité AAE sont décrites dans le Tableau 127.

Tableau 127 – Descriptions ServiceType()

Nom	ServiceType	Utilisée dans	AAE
Entrée		Sortie	
Données		Reçoit le type de primitive du message de service	
Fonction			
Cette fonction permet de déterminer le type du message de service reçu, notamment la primitive d'indication ConfirmedService, la primitive de confirmation ConfirmedService et la primitive d'indication UnconfirmedService			

9.3 Diagramme d'états de protocole de l'élément ASE d'événement

9.3.1 Descriptions des états

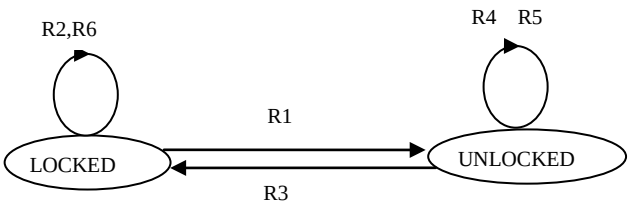
L'élément ASE d'événement possède deux états: LOCKED et UNLOCKED sont décrits dans le Tableau 128.

Tableau 128 – Etats de l'élément ASE d'événement

Etats	Descriptions
LOCKED	LOCKED(Enabled = FALSE) signifie que le système n'effectue pas de transfert de notification d'événements
UNLOCKED	UNLOCKED(Enable = TRUE) signifie que le système effectue un transfert de notification d'événements normal

9.3.2 Transitions d'états

Les transitions d'états de l'élément ASE d'événement sont présentées dans la Figure 6 et Tableau 129.



Légende

Anglais	Français
LOCKED	VERROUILLE
UNLOCKED	DEVERROUILLE

Figure 6 – Diagramme de transitions d'états de l'élément ASE d'événement

Tableau 129 – Table de transitions d'états de l'élément ASE d'événement

#	Etat actuel	Evénement ou condition => action	Etat suivant

R1	LOCKED	ReportConditionChanging.ind
		&& Enabled = TRUE
		=>
		ReportConditionChanging.rsq(+){
		}
			UNLOCKED

R2	LOCKED	ReportConditionChanging.ind
		&&Enabled = FALSE
		=>
		ReportConditionChanging.rsq(+){
		}
			LOCKED

R3	UNLOCKED	ReportConditionChanging.ind
		&&Enabled = FALSE
		=>
		ReportConditionChanging.rsq(+){
		}
			LOCKED

R4	UNLOCKED	ReportConditionChanging.ind
		&&Enabled = TRUE
		=>
		ReportConditionChanging.rsq(+){
		}
			UNLOCKED

R5	UNLOCKED	Type 14 AL service.ind <>" ReportConditionChanging.ind" => (aucune action entreprise)	UNLOCKED
R6	LOCKED	Type 14 AL service.ind <>" ReportConditionChanging.ind" => (aucune action entreprise)	LOCKED

9.3.3 Descriptions des fonctions

Les passages d'état de l'élément ASE d'événement n'utilisent pas d'autres fonctions.

9.4 Diagramme d'états de protocole de l'élément ASE de domaine

9.4.1 Description des états

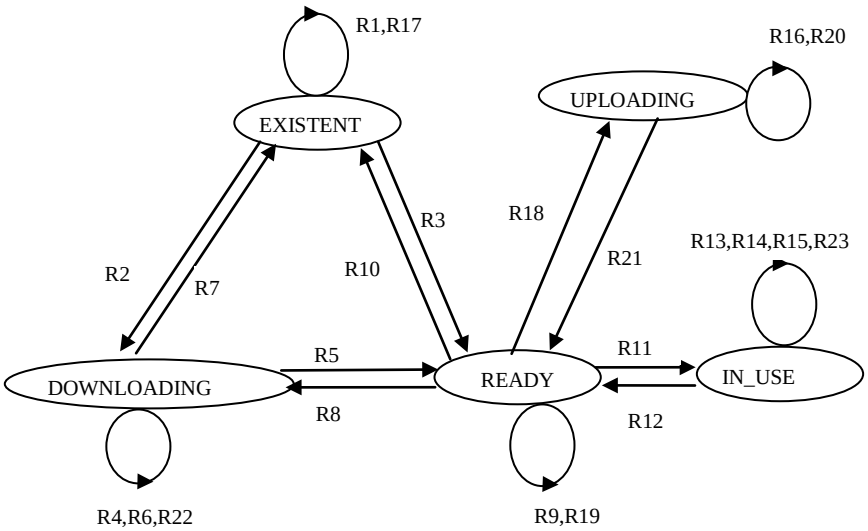
L'élément ASE de domaine possède cinq états décrits dans le Tableau 130.

Tableau 130 – Etats de l'élément ASE de domaine

Code d'état du domaine	réelle	Spécification
EXISTENT	0	Ce domaine existe, mais son contenu n'est pas défini
DOWNLOADING	1	Ce domaine est en cours de téléchargement
UPLOADING	2	Ce domaine est en cours de chargement
READY	3	Ce domaine peut être utilisé
IN-USE	4	Ce domaine est en cours d'utilisation par un programme. Il ne peut pas être téléchargé ni être chargé tant qu'il est dans cet état

9.4.2 Transitions d'états

Les passages d'état de l'élément ASE de domaine sont présentés dans la Figure 7 et le Tableau 131.



Légende

Anglais	Français
EXISTENT	EXISTANT
UPLOADING	EN TELECHARGEMENT
DOWNLOADING	EN TELECHARGEMENT
READY	PRET
IN USE	UTILISE

Figure 7 – Diagramme de transitions d'états de l'élément ASE de domaine

Tableau 131 – Table de transitions d'états de l'élément ASE de domaine

	Etat actuel	Evénement ou condition => action	Etat suivant

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

	Etat actuel	Evénement ou condition => action	Etat suivant
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	EN COURS D'UTILISATION
R12	EN COURS D'UTILISATION	DecreamentInvokeDomainCounter() && Counter=1 => Counter = 0	READY
R13	EN COURS D'UTILISATION	IncreamentInvokeDomainCounter() => Counter = Counter +1	EN COURS D'UTILISATION
R14	EN COURS D'UTILISATION	DecreamentInvokeDomainCounter() &&counter>1 => Counter = Counter -1	EN COURS D'UTILISATION

R15	EN COURS D'UTILISATION	DomainDownload.ind
		=>
		DomainDownload.err{
		}
			EN COURS D'UTILISATION

R16	UPLOADING	DomainDownload.ind	UPLOADING

	Etat actuel	Evénement ou condition => action	Etat suivant

		=>
		DomainDownload.rsp(-){
		ErrorType:=Service Error
		}	

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	EN COURS D'UTILISATION	DomainUpload.ind
		=>
		DomainUpload.rsp (-){
		ErrorType:=Service Error
		}
			EN COURS D'UTILISATION

9.4.3 Descriptions des fonctions

Les fonctions utilisées par les passages d'état de l'élément ASE de domaine sont décrites dans les tableaux qui suivent (Tableau 132 à Tableau 135).

9.4.3.1 Description de la fonction Domain_DownloadSucceed()

La fonction Domain_DownloadSucceed() est décrite dans le Tableau 132.

Tableau 132 – Description Domain_DownloadSucceed()

Nom	Domain_DownloadSucceed	Utilisée dans	AAE
Entrée		Sortie	
Aucun		TRUE ou FALSE	
Fonction			
Cette fonction permet de déterminer l'état de téléchargement. Elle renvoie la valeur FALSE si l'opération échoue ou TRUE si l'opération réussit			

9.4.3.2 Description de la fonction Domain_WriteBuffer()

La fonction Domain_WriteBuffer() est décrite dans le Tableau 133.

Tableau 133 – Description Domain_WriteBuffer()

Nom	Domain_WriteBuffer	Utilisée dans	AAE
Entrée		Sortie	
Aucun		Aucun	
Fonction			
Ecrire les données reçues dans le tampon			

9.4.3.3 Description de la fonction IncreamentInvokeDomainCounter()

La fonction IncreamentInvokeDomainCounter() est décrite dans le Tableau 134.

Tableau 134 – Description IncreamentInvokeDomainCounter()

Nom	IncreamentInvokeDomainCounter	Utilisée dans	AAE
Entrée		Sortie	
Aucun		Aucun	
Fonction			
La fonction IncreamentInvokeDomainCounter est incrémentée de la valeur 1			

9.4.3.4 Description de la fonction DecreamentInvokeDomainCounter()

La fonction DecreamentInvokeDomainCounter() est décrite dans le Tableau 135.

Tableau 135 – Description DecreamentInvokeDomainCounter()

Nom	DecreamentInvokeDomainCounter	Utilisée dans	AAE
Entrée		Sortie	
Aucun		Aucun	
Fonction			
La fonction IncreamentInvokeDomainCounter est incrémentée de la valeur 1			

9.5 Machine de protocole ASE de bloc

9.5.1 Descriptions des états

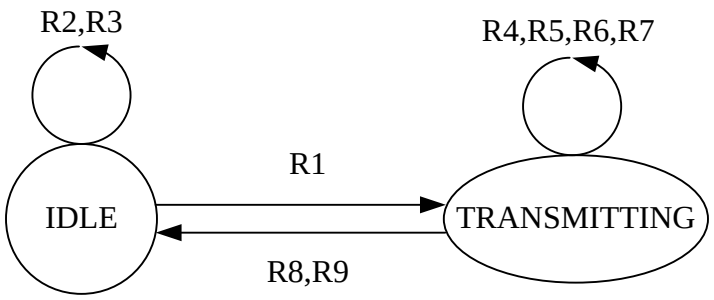
L'élément ASE de bloc possède deux états: IDLE et TRANSMISSION tel que décrit dans le Tableau 136.

Tableau 136 – Valeur d'état de la transmission de bloc

Etats	Descriptions
IDLE	L'appareil n'a actuellement aucune transmission
TRANSMITTING	L'appareil transmet des données de bloc

9.5.2 Transitions d'états

Les transitions d'état de Block ASE sont présentées à la Figure 8 et au Tableau 137.



Légende

Anglais	Français
IDLE	INTERROMPU
TRANSMITTING	EN TRANSMISSION

Figure 8 – Diagramme de transitions d'états de l'élément ASE de bloc

Tableau 137 – Table de transitions d'états de l'élément ASE de bloc

#	Etat actuel	Evénement ou condition => action	Etat suivant

R1	IDLE	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = TRUE
		&& MemberNum = 0
		=>
		BlockTransmissionOpen.rsp(+){
		}
		BlockTransmit.ind{
		}
		&& MemberNum = 1
			TRANSMITTING

R2	IDLE	BlockTransmissionClose.ind
		=>
		BlockTransmissionClose.rsp(-){
		ErrorType:=Service Error
		}
			IDLE

R3	IDLE	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = FALSE
		=>
		BlockTransmissionOpen.err{
		}
			IDLE

R4	TRANSMITTING	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = FALSE
		=>
		BlockTransmissionOpen.err{
		}
			TRANSMITTING

#	Etat actuel	Evénement ou condition => action	Etat suivant

R5	TRANSMITTING	BlockTransmissionOpen.ind
		&& BlockTransmissionOpenSucceed() = TRUE
		&& MemberNum > 0
		=>
		BlockTransmissionOpen.rsp(+){
		}
		MemberNum = MemberNum + 1
			TRANSMITTING

R6	TRANSMITTING	BlockTransmissionClose.ind
		&& BlockTransmissionCloseSucceed() = TRUE
		&& MemberNum > 1
		=>
		BlockTransmissionClose.rsp(+){
		}
		MemberNum = MemberNum – 1
			TRANSMITTING

R7	TRANSMITTING	BlockTransmissionClose.ind
		&& BlockTransmissionCloseSucceed() = FALSE
		=>
		BlockTransmissionClose.err{
		}
			TRANSMITTING

R8	TRANSMITTING	BlockTransmissionClose.ind
		&& BlockTransmissionCloseSucceed() = TRUE
		&& MemberNum = 1
		=>
		BlockTransmissionClose.rsp(+){
		}
		&& MemberNum = 0
			IDLE

R9	TRANSMITTING	ReceiveBlockTransmissionHeartbeat_timeout = TRUE => ReceiveBlockTransmissionHeartbeat_timeout() && MemberNum = 0	IDLE

9.5.3 Descriptions des fonctions

Du Tableau 138 au Tableau 140 sont décrites les fonctions utilisées par les transitions d'état Block ASE.

Tableau 138 – Descriptions BlockTransmissionOpenSucceed()

Nom	BlockTransmissionOpenSucceed	Utilisée dans	AAE
Entrée		Sortie	
Données		TRUE ou FALSE	
Fonction			
Détermine l'état BlockTransmissionOpen, en cas d'échec, puis renvoie FALSE; en cas de réussite, renvoie TRUE			

Tableau 139 – Descriptions BlockTransmissionCloseSucceed()

Nom	BlockTransmissionCloseSucceed	Utilisée dans	AAE
Entrée		Sortie	
Données		TRUE ou FALSE	
Fonction			
Détermine l'état BlockTransmissionClose, en cas d'échec, puis renvoie FALSE; en cas de réussite, renvoie TRUE			

Tableau 140 – Descriptions ReceiveBlockTransmissionHeartbeat_timeout()

Nom	ReceiveBlockTransmissionHeartbeat_timeout	Utilisée dans	AAE
Entrée		Sortie	
Aucun		Aucun	
Fonction			
Cette fonction met fin à la transmission des données de bloc si l'appareil ne reçoit pas le service BlockTransmissionHeartbeat dans le délai imparti			

10 Diagramme d'états de relations d'applications

10.1 Primitives

10.1.1 Primitives échangées entre AREP et FME(ou AAE)

Le Tableau 141 et le Tableau 142 décrivent les primitives échangées entre le point d'extrémité AREP et l'entité FME (ou AAE).

Tableau 141 – Primitives adressées par FME (ou AAE) à AREP

Nom de la primitive	Source	Paramètres associés	Fonctions
DTC_req	FME (ou AAE)	remote_ip_address, données	Cette primitive permet à l'entité FME (ou AAE) de transmettre une primitive de demande ConfirmedService au point de fin AREP
DTC_rsp	FME (ou AAE)	remote_ip_address, données	Cette primitive permet à l'entité FME (ou AAE) de transmettre une primitive de réponse ConfirmedService au point de fin AREP
DTU_req	FME (ou AAE)	remote_ip_address, données	Cette primitive permet à l'entité FME (ou AAE) de transmettre une primitive de demande UnconfirmedService au point de fin AREP

Tableau 142 – Primitives adressées par AREP à FME (ou AAE)

Nom de la primitive	Source	Paramètres associés	Fonctions
DTC_ind	AREP	remote_ip_address, données	Cette primitive permet au point de fin AREP de transmettre une primitive d'indication ConfirmedService à l'entité FME (ou AAE)
DTC_cnf	AREP	remote_ip_address, données	Cette primitive permet au point de fin AREP de transmettre une primitive de confirmation ConfirmedService à l'entité FME (ou AAE)
DTU_ind	AREP	remote_ip_address, données	Cette primitive permet au point de fin AREP de transmettre une primitive d'indication UnconfirmedService à l'entité FME (ou AAE)

10.1.2 Paramètres de primitives échangés AREP et FME (ou AAE)

Le Tableau 143 décrit les paramètres utilisés dans les primitives échangées entre AREP et FME (ou AAE).

Tableau 143 – Paramètres de primitives échangés entre AREP et FME (ou AAE)

Nom de paramètre	Description
remote_ip_address	Ce paramètre transfère l'adresse IP de l'appareil distant, c'est-à-dire l'adresse de destination à laquelle l'expéditeur envoie des données et l'adresse source des données reçues par le destinataire
Données	Ce paramètre transfère les données envoyées par l'expéditeur ainsi que les données reçues par le destinataire

10.1.3 Primitives échangées entre AREP et ESME

Les primitives échangées entre AREP et ESME sont illustrées dans le Tableau 144 et le Tableau 145.

Tableau 144 – Primitives adressées par AREP à ESME

Nom de la primitive	Source	Paramètres associés	Fonctions
Type 14_PDU_req	AREP	remote_ip_address, données	Cette primitive permet à l'entité AAE de transmettre une primitive de demande ConfirmedService à l'entité ESME

Tableau 145 – Primitives adressées par ESME à AREP

Nom de la primitive	Source	Paramètres associés	Fonctions
Type 14_PDU_ind	ESME	remote_ip_address, données	Cette primitive permet à l'entité ESME de transmettre une primitive d'indication ConfirmedService au point de fin AREP

10.1.4 Paramètres de primitives échangés entre AREP et ESME

Le Tableau 146 décrit les paramètres utilisés dans les primitives échangées entre le point de fin AAE et l'entité ESME.

Tableau 146 – Paramètres de primitives échangés entre AREP et ESME

Nom de paramètre	Description
remote_ip_address	Ce paramètre transfère l'adresse IP de l'appareil distant, c'est-à-dire l'adresse de destination à laquelle l'expéditeur envoie des données et l'adresse source des données reçues par le destinataire
Données	Ce paramètre transfère les données envoyées par l'expéditeur ainsi que les données reçues par le destinataire

10.2 Descriptions des états AREP

Le point d'extrémité AREP de type 14 est toujours actif. Son état est décrit dans le Tableau 147.

Tableau 147 – Descriptions des états AREP

Etats	Descriptions
ACTIVE	Les points de fin AREP à l'état ACTIF sont prêts à transférer des primitives de service à l'utilisateur ALU et à l'entité ESME, ou à recevoir des primitives de l'utilisateur ALU et de l'entité ESME

10.3 Transitions d'états

Le diagramme d'états protocolaire du point d'extrémité AREP est illustré dans la Figure 9 et au Tableau 148.

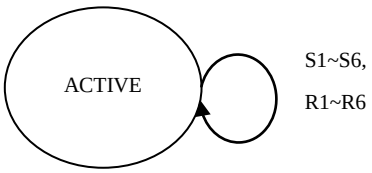


Figure 9 – Diagramme de transitions d'états du point d'extrémité AREP

Tableau 148 – Transitions des états AREP

#	Etat actuel	Evénement ou condition => action	Etat suivant

S1	ACTIVE	DTC_req
		|| DTC_rsp
		|| DTU_req
		=>
		Type 14_PDU_req {
		user_data:= Data,
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

R1	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = "peer"
		&& MessageType(data) = "Confirmed Service Indication"
		=>
		DTC_cnf {
		Data:= user_data
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

R2	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = "peer"
		&& MessageType(data) = "Confirmed Service Confirmation"
		=>
		DTC_cnf {
		Data:= user_data
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

R3	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = "client"
		=>
		DTC_cnf {
		Data:= user_data
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

R4	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = "server"
		=>
		DTC_ind{
		Data:= user_data
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

R5	ACTIVE	Type 14_PDU_ind && AREPType(data) = "publisher" => Aucune action	ACTIVE

R6	ACTIVE	Type 14_PDU_ind
		&& AREPType(data) = "subscriber"
		=>
		DTU_ind{
		Data:= user_data
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

10.4 Descriptions des fonctions

Le Tableau 149 fonctions utilisées par les transitions d'états AREP.

Tableau 149 – Descriptions AREPType()

Nom	AREPType	Utilisée dans	AREP
Entrée		Sortie	
Données		Type d'AREP	
Fonction			
Cette fonction permet de déterminer le type de l'extrémité AREP, notamment l'homologue, le client, le serveur, le mode de publication et d'abonnement			

Les fonctions utilisées par les passages d'état de l'entité AAE sont décrites dans le Tableau 150.

Tableau 150 – Descriptions ServiceType()

Nom	ServiceType	Utilisée dans	AAE
Entrée		Sortie	
Données		Reçoit le type de primitive du message de service	
Fonction			
Cette fonction permet de déterminer le type du message de service reçu, notamment la primitive d'indication ConfirmedService, la primitive de confirmation ConfirmedService et la primitive d'indication UnconfirmedService			

11 Diagramme protocolaire de mapping de couche DLL

11.1 Concept

Dans un appareil de Type 14, les protocoles UPD/IP et ISO/CEI 8802-3 s'appliquent directement au système de type 14 en tant que sous-couches de la couche DLL définie dans la CEI 61158-1. Cet article définit l'interface entre les services de couche FAL de type 14 et la couche UDP/IP appelée Socket Mapping Entity (ESME) de type 14.

11.2 Primitives

11.2.1 Primitives et paramètres échangés entre l'entité AAE et l'entité ESME

Les primitives échangées entre l'entité AAE et l'entité ESME sont décrites en 9.1.3.

Les paramètres utilisés dans les primitives échangées entre l'entité AAE et l'entité ESME sont décrits en 9.1.4.

11.2.2 Primitives et paramètres échangés entre l'entité FME et l'entité ESME

Les primitives échangées entre l'entité FME et l'entité ESME sont décrites en 8.1.3.

Les paramètres utilisés dans les primitives échangées entre l'entité FME et l'entité ESME sont décrites en 8.1.4.

11.2.3 Primitives échangées entre la couche Transport et l'entité ESME

Le Tableau 151 décrit les primitives échangées entre la couche Transport (UDP) et l'entité ESME.

Tableau 151 – Primitives échangées entre la couche Transport et l'entité ESME

Nom de la primitive	source	Paramètres référencés
Type 14_PDU_req	Socket Mapping Entity	remote_ip_address, données
Type 14_PDU_ind	Couche Transport	remote_ip_address, données

11.2.4 Paramètres de primitives échangés entre la couche Transport et l'entité ESME

Le Tableau 152 décrit les paramètres de primitives échangés entre la couche Transport et l'entité ESME.

Tableau 152 – Paramètres de primitives échangés entre la couche Transport et l'entité ESME

Nom de paramètre	description
remote_ip_address	Ce paramètre transfère l'adresse IP de l'appareil distant, c'est-à-dire l'adresse de destination à laquelle l'expéditeur envoie des données et l'adresse source des données reçues par le destinataire
Données	Ce paramètre transfère les données envoyées par l'expéditeur ainsi que les données reçues par le destinataire

11.3 Descriptions des états

L'entité ESME est toujours active. Son état est décrit dans le Tableau 153.

Tableau 153 – Descriptions des états de l'entité ESME

Nom de l'état	description
ACTIVE	L'entité ESME transfère des primitives à l'entité AAE ou à l'entité FME pour transmission à la couche de transport; ou est prête à recevoir des primitives envoyées par l'entité AAE, ou l'entité FME à la couche de transport

11.4 Transitions d'états

La Figure 10 et le Tableau 154 illustrent les passages d'état de l'entité ESME.

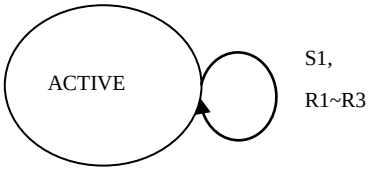


Figure 10 – Diagramme de transitions d'états de l'entité ESME

Tableau 154 – Passages d'état de l'entité ECFME

#	Etat actuel	Evénement ou condition => action	Etat suivant

	ACTIVE	DTC_req
		|| DTC_rsp
		|| DTU_req
		=>
		Type 14-PDU.req {
		Senddata := data,
		Destination_ip: = remote_ip_address,
		}
			ACTIVE

	ACTIVE	Type 14-PDU.ind
		&& ServiceType(data) = "Confirmed Service Indication"
		=>
		DTC.ind{
		Receivedata:= data,
		Remote_ip: = remote_ip_address,
		}
			ACTIVE

	ACTIVE	Type 14-PDU.ind
		&& ServiceType(data) = "Confirmed Service Confirmation"
		=>
		DTC.cnf{
		Receivedata:= data,
		Remote_ip: = remote_ip_address,
		}
			ACTIVE

	ACTIVE	Type 14-PDU.ind
		&& ServiceType(data) = "Unconfirmed Service Indication"
		=>
		DTU.ind{
		Receivedata:= data,
		Remote_ip: = remote_ip_address,
		}
			ACTIVE

11.5 Descriptions des fonctions

La fonction utilisée par les passages d'état de l'entité ESME est décrite dans le Tableau 155.

Tableau 155 – ServiceType()description

nom	ServiceType	utilisation	Socket Mapping Entity
entrée		sortie	
Données		Reçoit le type de primitive du message de service	
fonction			
Cette fonction permet de déterminer le type du message de service reçu, notamment la primitive d'indication ConfirmedService, la primitive de confirmation ConfirmedService et la primitive d'indication UnconfirmedService			

Bibliographie

CEI 61158-1, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Présentation et lignes directrices des séries CEI 61158 et CEI 61784*

CEI 61784-1, *Réseaux de communication industriels – Profils – Partie 1: Profils de bus de terrain*

CEI 61784-2, *Réseaux de communication industriels – Profils – Partie 2: Profils de bus de terrain supplémentaires pour les réseaux en temps réel basés sur l'ISO/CEI 8802-3*

ISO/IEC/TR 8802-1, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 1: Overview of Local Area Network Standards* (disponible en anglais seulement)

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

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1* (disponible en anglais seulement)

ISO/IEC 8877, *Information technology – Telecommunications and information exchange between systems – Interface connector and contact assignments for ISDN Basic Access Interface located at reference points S and T* (disponible en anglais seulement)

ISO 8601, *Éléments de données et formats d'échange – Échange d'information – Représentation de la date et de l'heure*

IEEE 802.1Q, *IEEE standard for Local and metropolitan area networks – Virtual bridged local area networks,* disponible à l'adresse <http://www.ieee.org>

_____

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch