

INTERNATIONAL STANDARD

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

Withhold



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2010 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.

If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

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

About the IEC

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

About IEC publications

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

- Catalogue of IEC publications: www.iec.ch/searchpub

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 It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

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

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

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

INTERNATIONAL STANDARD

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

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE **XC**

ICS 25.04.40; 35.100.70; 35.110

ISBN 978-2-88912-130-4

CONTENTS

FOREWORD.....	7
INTRODUCTION.....	9
1 Scope.....	10
1.1 General.....	10
1.2 Specifications.....	10
1.3 Conformance.....	11
2 Normative references	11
3 Terms, definitions, symbols, abbreviations and conventions	11
3.1 Referenced terms and definitions	11
3.2 Fieldbus application layer specific terms and definitions	13
3.3 Abbreviations and symbols.....	16
3.4 Conventions	17
4 Abstract syntax.....	18
4.1 Fixed format PDU description	18
4.2 Object definitions in FAL management ASE.....	26
4.3 Definition of objects used in Type 14 application access entity.....	33
5 Transfer syntax	36
5.1 Encoding of basic data types.....	36
5.2 Encoding of Type 14 APDU header.....	43
5.3 Encoding of FAL management entity service parameters.....	43
5.4 Encoding of AAE Services.....	49
6 Structure of FAL protocol state machines	55
7 AP-Context state machine.....	56
7.1 Primitives exchanged between ALU and ALE	56
7.2 Protocol state machine descriptions	56
7.3 State transitions	57
7.4 Function descriptions	63
8 FAL management state machines.....	63
8.1 Primitives	63
8.2 Protocol state machine descriptions	65
8.3 State transitions	65
8.4 Function descriptions	67
9 Application access entity protocol machine.....	71
9.1 Primitives	71
9.2 AAE state machine.....	73
9.3 Event ASE protocol machine	76
9.4 Domain ASE protocol machine	77
10 Application relationship state machine.....	81
10.1 Primitives	81
10.2 AREP state description.....	82
10.3 State transitions	83
10.4 Function descriptions	84
11 DLL mapping protocol machine	84
11.1 Concept	84
11.2 Primitives	84

11.3 State description	85
11.4 State transitions	85
11.5 Function description	86
Bibliography.....	87
Figure 1 – State transition diagram	17
Figure 2 – Exchanged primitives of protocol state machine	56
Figure 3 – ACE protocol state machine	57
Figure 4 – FME protocol state machine	65
Figure 5 – AAE state transition diagrams	73
Figure 6 – Event ASE state transition diagrams	76
Figure 7 – Domain ASE state transition diagram	78
Figure 8 – AREP state transition diagrams.....	83
Figure 9 – ESME state transition.....	85
Table 1 – State machine description elements	18
Table 2 – Definition of Type 14 MOB header object.....	26
Table 3 – Definition of Type 14 device descriptor object	26
Table 4 – Definition of the time synchronization object.....	27
Table 5 – Definition of maximum response time object.....	28
Table 6 – Definition of the Type 14 communication scheduling management object	28
Table 7 – Definition of the device application information object	28
Table 8 – Definition of FB application information header.....	29
Table 9 – Definition of domain application information header.....	29
Table 10 – Definition of Type 14 link object header.....	30
Table 11 – Definition of Type 14 FRT link object header	30
Table 12 – Definition of FB application information object	31
Table 13 – Definition of Type 14 link object	31
Table 14 – Definition of Type 14 FRT link object.....	32
Table 15 – Definition of domain application information object	33
Table 16 – Definition of domain object	33
Table 17 – Definition of simple variable object	34
Table 18 – Definition of event object.....	34
Table 19 – Definition of Type 14 socket mapping object.....	35
Table 20 – Definition of Type 14 socket timer object	35
Table 21 – Definition of ErrorType object	36
Table 22 – Encoding of Boolean value TRUE.....	36
Table 23 – Encoding of Boolean value FALSE	36
Table 24 – Encoding of Unsigned8 data type	37
Table 25 – Encoding of Unsigned16 data type	37
Table 26 – Encoding of Unsigned32 data type	37
Table 27 – Encoding of Unsigned64 data type	38
Table 28 – Encoding of Int8 data type.....	38

Table 29 – Encoding of Int16 data type	38
Table 30 – Encoding of Int32 data type	39
Table 31 – Encoding of Int64 data type	39
Table 32 – Encoding of Real type	39
Table 33 – Encoding of VisibleString data type	40
Table 34 – Encoding of OctetString data type	40
Table 35 – Encoding of BitString data type	40
Table 36 – Encoding of TimeOfDay data type	41
Table 37 – Encoding of BinaryDate data type	42
Table 38 – Encoding of PrecisionTimeDifference data type	42
Table 39 – Encoding of Type 14 application layer service message header	43
Table 40 – Encoding of EM_DetectingDevice request parameters	43
Table 41 – Encoding of EM_OnlineReply request parameters	44
Table 42 – Encoding of EM_GetDeviceAttribute request parameters	44
Table 43 – Encoding of EM_GetDeviceAttribute positive response parameters	44
Table 44 – Encoding of EM_GetDeviceAttribute negative response parameters	45
Table 45 – Encoding of EM_ActiveNotification request parameters	46
Table 46 – Encoding of EM_ConfiguringDevice request parameters	47
Table 47 – Encoding of EM_ConfiguringDevice positive response parameters	48
Table 48 – Encoding of EM_ConfiguringDevice negative response parameters	48
Table 49 – Encoding of EM_SetDefaultValue request parameters	48
Table 50 – Encoding of EM_SetDefaultValue positive response parameters	48
Table 51 – Encoding of clear device attribute service refuse packet	49
Table 52 – Encoding of DomainDownload request parameters	49
Table 53 – Encoding of domain download service response packet	49
Table 54 – Encoding of DomainDownload negative response parameters	49
Table 55 – Encoding of DomainUpload request parameters	50
Table 56 – Encoding of DomainUpload positive response parameters	50
Table 57 – Encoding of DomainUpload negative response parameters	50
Table 58 – Encoding of EventReport request parameters	51
Table 59 – Encoding of EventReportAcknowledge request parameters	51
Table 60 – Encoding of EventReportAcknowledge positive response parameters	51
Table 61 – Encoding of EventReportAcknowledge negative response parameters	51
Table 62 – Encoding of ReportConditionChanging request parameters	52
Table 63 – Encoding of ReportConditionChanging positive response parameters	52
Table 64 – Encoding of ReportConditionChanging negative response parameters	52
Table 65 – Encoding of Read request parameters	52
Table 66 – Encoding of Read positive response parameters	53
Table 67 – Encoding of Read negative response parameters	53
Table 68 – Encoding of Write request parameters	53
Table 69 – Encoding of Write positive response parameters	53
Table 70 – Encoding of Write negative response parameters	54
Table 71 – Encoding of VariableDistribute request parameters	54

Table 72 – Encoding of FRTRead request parameters	54
Table 73 – Encoding of FRTRead positive response parameters.....	54
Table 74 – Encoding of FRTRead negative response parameters	54
Table 75 – Encoding of FRTWrite request parameters	55
Table 76 – Encoding of FRTWrite positive response parameters.....	55
Table 77 – Encoding of FRTWrite negative response parameters	55
Table 78 – Encoding of FRTVariableDistribute request parameters.....	55
Table 79 – Primitives delivered by ALU to ALE	56
Table 80 – Primitives delivered by ALE to ALU	56
Table 81 – ACE state descriptions	57
Table 82 – ACE state transitions (sender).....	57
Table 83 – ACE state transitions (receiver).....	60
Table 84 – APServiceType() descriptions.....	63
Table 85 – Primitives delivered by application layer user to FME	63
Table 86 – Primitives delivered by FME to application layer user	64
Table 87 – Primitive parameters exchanged between FME and application layer user	64
Table 88 – Primitives delivered by FME to ESME	64
Table 89 – Primitives delivered by ESME to FME	64
Table 90 – Primitives parameters exchanged between FME and ESME	65
Table 91 – State transitions of Type 14 FME.....	66
Table 92 – RcvNewIpAddress() descriptions.....	68
Table 93 – Attribute_Set() descriptions	68
Table 94 – RestoreDefaults() descriptions	68
Table 95 – NewAddress() descriptions.....	68
Table 96 – Restart_Type 14RepeatTimer() descriptions.....	68
Table 97 – Clear_DuplicatePdTagFlag() descriptions.....	69
Table 98 – Type 14RepeatTimerExpire() descriptions	69
Table 99 – Send_EM_ReqRspMessage() descriptions	69
Table 100 – Send_EM_CommonErrorRsp() descriptions.....	69
Table 101 – SntpSyncLost() descriptions	70
Table 102 – IPAddressCollision() descriptions	70
Table 103 – RecvMsg() descriptions	70
Table 104 – QueryMatch() descriptions.....	70
Table 105 – MessageIDMatch() descriptions.....	70
Table 106 – DevId_Match() descriptions	71
Table 107 – PdTag_Match() descriptions	71
Table 108 – Set_Attribute_Data() descriptions	71
Table 109 – Set_DuplicatePdTagFlag() descriptions	71
Table 110 – Primitives issued by ALU to AAE	72
Table 111 – Primitives issued by AAE to ALU	72
Table 112 – Primitives parameters exchanged between AAE and ALU.....	72
Table 113 – Primitives issued by AAE to ESME	72
Table 114 – Primitives issued by ESME to AAE	72

Table 115 – Primitive parameters exchanged between AAE and ESME	73
Table 116 – AAE state descriptions	73
Table 117 – AAE state transitions (sender)	74
Table 118 – AAE state transitions (receiver)	75
Table 119 – ServiceType() descriptions	75
Table 120 – State value of event management.....	76
Table 121 – Event ASE state transition table	77
Table 122 – Domain state value.....	77
Table 123 – Domain ASE state transition table	78
Table 124 – Domain_DownloadSucceed() description	80
Table 125 – Domain_WriteBuffer() description	81
Table 126 – IncrementInvokeDomainCounter() description	81
Table 127 – DecreamentInvokeDomainCounter() description	81
Table 128 – Primitives issued by FME(or AAE) to AREP	81
Table 129 – Primitives issued by AREP to FME(or AAE)	82
Table 130 – Primitives parameters exchanged between AREP and FME(or AAE)	82
Table 131 – Primitives issued by AREP to ESME	82
Table 132 – Primitives issued by ESME to AREP	82
Table 133 – Primitive parameters exchanged between AREP and ESME	82
Table 134 – AREP state descriptions	83
Table 135 – AREP state transitions.....	83
Table 136 – AREPType() descriptions.....	84
Table 137 – ServiceType() descriptions	84
Table 138 – The primitives exchanged between transport layer and ESME	85
Table 139 – Primitives parameters exchanged between Transport Layer and ESME	85
Table 140 – ESME state description	85
Table 141 – ECFME state transitions	86
Table 142 – ServiceType()description	86

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.

NOTE 1 Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in Type combinations as specified explicitly in the profile parts. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

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

This second edition cancels and replaces the first edition published in 2007. This edition constitutes a technical revision.

The main changes with respect to the previous edition are listed below:

- update the Normative references and Bibliography;
- corrections of editorial errors;

- specification changes for CPF3;
- 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/607/FDIS	65C/621/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.

NOTE 2 The revision of this standard will be synchronized with the other parts of the IEC 61158 series.

Withdrawn

INTRODUCTION

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

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

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

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

Withdrawn

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

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

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

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

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

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

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

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

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

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

- a) application entity
- b) application process

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

3.1.2 ISO/IEC 8822 terms

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

- a) abstract syntax
- b) presentation context

3.1.3 ISO/IEC 9545 terms

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

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

3.1.4 ISO/IEC 8824 terms

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

- a) object identifier
- b) type

3.1.5 Fieldbus data-link Layer terms

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

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

- l) node
- m) scheduled

3.2 Fieldbus application layer specific terms and definitions

3.2.1

access control

control on the reading and writing of an object

3.2.2

access Path

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

3.2.3

communication macrocycle

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

3.2.4

communication scheduling

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

3.2.5

configuration (of a system or device)

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

3.2.6

cyclic

repetitive in a regular manner

3.2.7

destination FB Instance

FB instance that receives the specified parameters

3.2.8

domain

part of memory used to store code or data

3.2.9

domain download

operation to write data in a domain

3.2.10

domain upload

operation to read data from a domain

3.2.11

entity

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

3.2.12

Type 14 bridge

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

3.2.13

identifier

16-bit word associated with a system variable

3.2.14

index

address of an object within an application process

3.2.15

instance

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

3.2.16

instantiation

creation of an instance of a specified type.

3.2.17

management information

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

3.2.18

management information base

organized list of management information

3.2.19

mapping

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

3.2.20

member

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

3.2.21

message filtering

decision on a message according to a special rule.

3.2.22

micro segment

part of a network, where special scheduling is implemented.

3.2.23

offset

number of octets from a specially designated position.

3.2.24

phase

elapsed fraction of a cycle, measured from some fixed origin

3.2.25

process interface

data exchange and information mapping between physical process and application unit

3.2.26

real-time

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

3.2.27**real-time communication**

transfer of data in real-time.

3.2.28**Real-Time Ethernet****RTE**

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

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

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

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

Withheld from

3.3 Abbreviations and symbols

AAE	Application Access Entity
AE	Application Entity
AL	Application Layer
ALE	Application Layer Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
ARP	Address Resolution Protocol
AREP	Application Relationship End Point
ASE	Application Service Element
Cnf	Confirmation
CR	Communication Relationship
CREP	Communication Relationship End Point
CSMA/CD	Carrier Sense Multiple Access Protocol with Collision Detection
DD	Device Description
DHCP	Dynamic Host Configuration Protocol
DL-	(as a prefix) data-link-
DLCEP	Data-link Connection End Point
DLL	Data-link Layer
DLE	Data-link Entity
DLM	Data-link management
DLS	Data-link Service
DLSAP	Data-link Service Access Point
DLSDU	DL-service data-unit
ECSME	Type 14 communication scheduling management entity
Type 14	Ethernet for Plant Automation
EM_	(as a prefix) Type 14 Management
ESME	Type 14 Socket Mapping Entity
FB	Function Block
FBAP	Function Block Application Process
FME	FAL Management Entity
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 state 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.

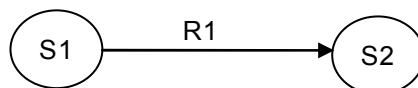


Figure 1 – State transition diagram

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
}

```

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-RequestPDU,
    FRTWrite [10]    IMPLICIT  FRTWrite-RequesPDU
}

```

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
}

```

}

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 {
    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 [4] IMPLICIT FRTVariableDistribute-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,
LANRedundancyPort	[6]	IMPLICIT	Unsigned16,
DeviceRedundancyState	[7]	IMPLICIT	Unsigned8,
MaxRedundancyNumber	[8]	IMPLICIT	Unsigned8,
ActiveIPAddress	[9]	IMPLICIT	Unsigned32

}

EM_ConfiguringDevice-ResponsePDU ::= CHOICE {

EM_ConfiguringDevice-PositiveResponsePDU,
EM_ConfiguringDevice-NegativeResponsePDU

}

EM_ConfiguringDevice-PositiveResponsePDU ::= SEQUENCE {

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

}

EM_ConfiguringDevice-NegativeResponsePDU ::= SEQUENCE {

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

}

4.1.9.6 EM_SetDefaultValue service

EM_SetDefaultValue-RequestPDU ::= SEQUENCE {

DestinationIPAddress	[0]	IMPLICIT	Unsigned32,
DeviceID	[1]	IMPLICIT	VisibleString,
PdTag	[2]	IMPLICIT	VisibleString

}

EM_SetDefaultValue-ResponsePDU ::= CHOICE {

EM_SetDefaultValue-PositiveResponsePDU ,
EM_SetDefaultValue-NegativeResponsePDU

}

EM_SetDefaultValue-PositiveResponsePDU ::= SEQUENCE {

DestinationIPAddress	[0]	IMPLICIT	Unsigned32
----------------------	-----	----------	------------

}

EM_SetDefaultValue-NegativeResponsePDU ::= SEQUENCE {

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

}

4.1.10 Application Access Entity (AAE) services

4.1.10.1 DomainDownload service

DomainDownload-RequestPDU ::= SEQUENCE {

SourceAppID	[0]	IMPLICIT	Unsigned16,
DestinationAppID	[1]	IMPLICIT	Unsigned16,
DestinationObjectID	[2]	IMPLICIT	Unsigned16,
DataNumber	[3]	IMPLICIT	Unsigned16,

```

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

```

4.1.10.2 DomainUpload service

```

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

```

4.1.10.3 EventReport service

```

    EventReport-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 AcknowledgeEventReport service

```

    AcknowledgeEventReport-RequestPDU ::= SEQUENCE {
        DestinationAppID      [0] IMPLICIT Unsigned16,
        DestinationObjectID    [1] IMPLICIT Unsigned16,
        EventNumber            [2] IMPLICIT Unsigned16
    }
    AcknowledgeEventReport-ResponsePDU ::= CHOICE {
        AcknowledgeEventReport-PositiveResponsePDU,
        AcknowledgeEventReport-NegativeResponsePDU
    }
    AcknowledgeEventReport-PositiveResponsePDU ::= SEQUENCE{

```

```

        DestinationAppID          [0]  IMPLICIT  Unsigned16
    }
    AcknowledgeEventReport -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
}

```

4.1.11 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)

Int16 ::= INTEGER (-32768..+32767)

-- integer range -27<= i <= 27-1

-- integer range -215<= i <= 215-1

Int32 ::= INTEGER
Int64 ::= INTEGER

-- integer range -231 <= i <= 231-1
-- integer range -263 <= i <= 263-1

4.1.11.3 Notation of unsigned integer type

Unsigned8 ::= INTEGER (0..255)
Unsigned16 ::= INTEGER (0..65535)
Unsigned32 ::= INTEGER
Unsigned64 ::= INTEGER

-- integer range 0 <= i <= 28-1
-- integer range 0 <= i <= 216-1
-- integer range 0 <= i <= 232-1
-- 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