IEC 61158-5-2

Edition 4.0 2019-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Industrial communication networks – Fieldbus specifications – Part 5-2: Application layer service definition – Type 2 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-2: Définition des services de la couche application – Eléments de type 2

IEC 61158-5-2:2019-04(en-fr)

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

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

IEC 61158-5-2

Edition 4.0 2019-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Industrial communication networks – Fieldbus specifications – Part 5-2: Application layer service definition – Type 2 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-2: Définition des services de la couche application – Eléments de type 2

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-9264-8

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-2: Application layer service definition – Type 2 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 61158-5-2 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This fourth edition cancels and replaces the third edition published in 2014. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

- addition of a data type in 5.3.2;
- clarifications of Object management ASE in 6.2.1;
- extensions of General ASE in 6.2.1.2.1;
- extensions/clarifications of Identity ASE in 6.2.1.2.2;
- update of Message Router ASE in 6.2.1.2.4;
- extensions/clarifications of Time Sync ASE in 6.2.1.2.6;
- updates of Parameter ASE in 6.2.1.2.7;
- updates of FAL ASE service specification in 6.2.1.3;
- extensions/clarifications of Connection manager ASE in 6.2.2;
- extensions/clarifications of Connection ASE in 6.2.3;
- extensions/clarifications of Application type in 6.3.1.4.5.
- miscellaneous editorial corrections.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
65C/947/FDIS	65C/950/RVD

Full information on the voting for the approval of this International 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.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

This document 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 service is provided by the application protocol making use of the services available from the data-link or other immediately lower layer. This document defines the application service characteristics that fieldbus applications and/or system management may exploit.

Throughout the set of fieldbus standards, the term “service” refers to the abstract capability provided by one layer of the OSI Basic Reference Model to the layer immediately above. Thus, the application layer service defined in this document is a conceptual architectural service, independent of administrative and implementation divisions.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-2: Application layer service definition – Type 2 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 part of IEC 61158 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 2 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 International Standard defines in an abstract way the externally visible service provided by the Type 2 fieldbus application layer in terms of:

a) an abstract model for defining application resources (objects) capable of being manipulated by users via the use of the FAL service,

b) the primitive actions and events of the service;

c) the parameters associated with each primitive action and event, and the form which they take; and

d) the interrelationship between these actions and events, and their valid sequences.

The purpose of this document is to define the services provided to:

a) the FAL user at the boundary between the user and the application layer of the fieldbus reference model, and

b) Systems Management at the boundary between the application layer and Systems Management of the fieldbus reference model.

This document specifies the structure and services of the Type 2 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI application layer structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application-entities (AE) contained within the application processes. The FAL AE is composed of a set of object-oriented application service elements (ASEs) and a layer management entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for the management of the instances of FAL classes.

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the applications are not specified; only a definition of what requests and responses they can

send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this document to provide access to the FAL to control certain aspects of its operation.

1.2 Specifications

The principal objective of this document is to specify the characteristics of conceptual application layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of application layer protocols for time-critical communications.

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 services standardized as the various Types of IEC 61158, and the corresponding protocols standardized in subparts of IEC 61158-6.

This specification may be used as the basis for formal application programming interfaces. Nevertheless, it is not a formal programming interface, and any such interface will need to address implementation issues not covered by this specification, including

a) the sizes and octet ordering of various multi-octet service parameters, and

b) the correlation of paired request and confirm, or indication and response, primitives.

1.3 Conformance

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

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill the Type 2 application layer services as defined in this document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

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 61131-3:2003[1], *Programmable controllers – Part 3: Programming languages*

IEC 61158-1:2019, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-2:2014, *Industrial communication networks – Fieldbus specifications – Part 3-2: Data-link layer service definition – Type 2 elements*

IEC 61158-3-2:2014/AMD1:2019

IEC 61158-4-2:2019, *Industrial communication networks – Fieldbus specifications – Part 4-2: Data-link layer protocol specification – Type 2 elements*

[1] A newer edition of this standard has been published, but only the cited edition applies.

IEC 61158-6-2:2019, *Industrial communication networks – Fieldbus specifications – Part 6-2: Application layer protocol specification – Type 2 elements*

IEC 61588:2009, *Precision clock synchronization protocol for networked measurement and control systems*

IEC 61784-3-2, *Industrial communication networks – Profiles – Part 3-2: Functional safety fieldbuses – Additional specifications for CPF 2*

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

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

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

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

ISO/IEC 10646, *Information technology – Universal Coded Character Set (UCS)*

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*

ISO 639-2, *Codes for the representation of names of languages – Part 2: Alpha-3 code*

ISO 8859-1[2]:1987, *Information processing – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO 8859-2[3]:1987, *Information processing – 8-bit single-byte coded graphic character sets – Part 2: Latin alphabet No. 2*

ISO 8859-3[4]:1988, *Information processing – 8-bit single-byte coded graphic character sets – Part 3: Latin alphabet No. 3*

ISO 8859-4[5]:1988, *Information processing – 8-bit single-byte coded graphic character sets – Part 4: Latin alphabet No. 4*

ISO 8859-5[6]:1988, *Information processing – 8-bit single-byte coded graphic character sets – Part 5: Latin/Cyrillic alphabet*

---

2 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

3 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

4 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

5 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

ISO 8859-6[7]:1987, *Information processing – 8-bit single-byte coded graphic character sets – Part 6: Latin/Arabic alphabet*

ISO 8859-7[8]:1987, *Information processing – 8-bit single-byte coded graphic character sets – Part 7: Latin/Greek alphabet*

ISO 8859-8[9]:1988, *Information processing – 8-bit single-byte coded graphic character sets – Part 8: Latin/Hebrew alphabet*

ISO 8859-9[10]:1989, *Information processing – 8-bit single-byte coded graphic character sets – Part 9: Latin alphabet No. 5*

ISO 11898:1993[11], *Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication*

IETF RFC 1759, *Printer MIB,* available at <http://www.ietf.org> [viewed 2018-09-04]

3 Terms, definitions, symbols, abbreviated terms and conventions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

3.1 ISO/IEC 7498-1 terms

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

---

6 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

7 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

8 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

9 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

10 A newer edition of this standard has been published by ISO/IEC, but the cited edition is the one used in the referenced IETF standards.

11 A newer edition of this standard has been published, but only the cited edition applies.

3.2 ISO/IEC 8822 terms

a) abstract syntax
b) presentation context

3.3 ISO/IEC 9545 terms

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

3.4 ISO/IEC 8824-1 terms

a) object identifier
b) type

3.5 Type 2 fieldbus data-link layer terms

The following terms, defined in IEC 61158-3-2 and IEC 61158-4-2, apply.

a) DL-time
b) DL-scheduling-policy
c) DLCEP
d) DLC
e) DL-connection-oriented mode
f) DLPDU
g) DLSDU
h) DLSAP
i) fixed tag
j) generic tag
k) link
l) MAC ID
m) network address
n) node address
o) node
p) tag
q) scheduled
r) unscheduled

3.6 Type 2 fieldbus application-layer specific definitions

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

3.6.1
allocate
take a resource from a common area and assign that resource for the exclusive use of a specific entity

3.6.2
application
function or data structure for which data is consumed or produced

3.6.3
application objects
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

3.6.4
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.6.5
application process object
component of an application process that is identifiable and accessible through an FAL application relationship

3.6.6
application process object class
class of application process objects defined in terms of the set of their network-accessible attributes and services

3.6.7
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

Note 1 to entry: This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities.

3.6.8
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.6.9
application relationship endpoint
context and behavior of an application relationship as seen and maintained by one of the application processes involved in the application relationship

Note 1 to entry: Each application process involved in the application relationship maintains its own application relationship endpoint.

3.6.10
attribute
description of an externally visible characteristic or feature of an object

Note 1 to entry: The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behavior of an object. Attributes are divided into class attributes and instance attributes.

3.6.11
behavior
indication of how an object responds to particular events

3.6.12
Best Master Clock Algorithm
BMCA
algorithm performed by each node to determine the clock that will become the master clock on a subnet and the grandmaster clock for the domain

Note 1 to entry: The algorithm primarily compares priority1, clock quality, priority2, and source identity to determine the best master among available candidates.

3.6.13
boundary clock
clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain

Note 1 to entry: It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock.

[SOURCE: IEC 61588:2009, 3.1.3, modified – second sentence changed to a Note]

3.6.14
class
set of objects, all of which represent the same kind of system component

Note 1 to entry: A class is a generalization of an object; a template for defining variables and methods. All objects in a class are identical in form and behavior, but usually contain different data in their attributes.

3.6.15
class attribute
attribute that is shared by all objects within the same class

3.6.16
class code
unique identifier assigned to each object class

3.6.17
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service

Note 1 to entry: A class specific object is unique to the object class which defines it.

3.6.18
client
a) object which uses the services of another (server) object to perform a task

b) initiator of a message to which a server reacts

3.6.19
clock
node participating in the Precision Time Protocol (PTP) that is capable of providing a measurement of the passage of time since a defined epoch

Note 1 to entry: There are three types of clocks in IEC 61588:2009, boundary, transparent and ordinary clocks.

[SOURCE: IEC 61588:2009, 3.1.4, modified – different Note]

3.6.20
communication objects
components that manage and provide a run time exchange of messages across the network

EXAMPLES Connection Manager object, Unconnected Message Manager (UCMM) object, and Message Router object.

3.6.21
connection
logical binding between application objects that may be within the same or different devices

Note 1 to entry: Connections may be either point-to-point or multipoint.

3.6.22
connection ID
CID
identifier assigned to a transmission that is associated with a particular connection between producers and consumers, providing a name for a specific piece of application information

3.6.23
connection path
octet stream that defines the application object to which a connection instance applies

3.6.24
connection point
buffer which is represented as a subinstance of an Assembly object

3.6.25
consume
act of receiving data from a producer

3.6.26
consumer
node or sink that is receiving data from a producer

3.6.27
consuming application
application that consumes data

3.6.28
cyclic
repetitive in a regular manner

3.6.29
device
physical hardware connected to the link

Note 1 to entry: A device may contain more than one node.

3.6.30
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.6.31
domain
logical grouping of clocks that synchronize to each other using the protocol, but that are not necessarily synchronized to clocks in another domain

[SOURCE: IEC 61588:2009, 3.1.7]

3.6.32
end node
producing or consuming node

3.6.33
endpoint
one of the communicating entities involved in a connection

3.6.34
epoch
origin of a time scale

[SOURCE: IEC 61588:2009, 3.1.9]

3.6.35
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.6.36
event
instance of a change of conditions

3.6.37
frame
denigrated synonym for DLPDU

3.6.38
grandmaster clock
within a domain, clock that is the ultimate source of time for clock synchronization using the PTP protocol

[SOURCE: IEC 61588:2009, 3.1.13]

3.6.39
group
a) <general> a general term for a collection of objects. Specific uses:

b) <addressing> when describing an address, an address that identifies more than one entity

3.6.40
interface
(a) shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

(b) collection of FAL class attributes and services that represents a specific view on the FAL class

3.6.41
invocation
act of using a service or other resource of an application process

Note 1 to entry: Each invocation represents a separate thread of control that may be described by its context. Once the service completes, or use of the resource is released, the invocation ceases to exist. For service invocations, a service that has been initiated but not yet completed is referred to as an outstanding service invocation. Also for service invocations, an Invoke ID may be used to unambiguously identify the service invocation and differentiate it from other outstanding service invocations.

3.6.42
instance
actual physical occurrence of an object within a class that identifies one of many objects within the same object class

EXAMPLE California is an instance of the object class state.

Note 1 to entry: The terms object, instance, and object instance are used to refer to a specific instance.

3.6.43
instance attribute
attribute whose value is unique to an object instance and whose definition is shared by all instances of an object

3.6.44
instantiated
object that has been created in a device

3.6.45
logical device
a certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.6.46
Lpacket
Link packet
piece of application information that contains a size, control octet, tag, and link data

Note 1 to entry: Peer data-link layers use Lpackets to send and receive service data units from higher layers in the OSI stack.

3.6.47
management information
network-accessible information that supports managing the operation of the fieldbus system, including the application layer

Note 1 to entry: Managing includes functions such as controlling, monitoring, and diagnosing.

3.6.48
master clock
in the context of a single Precision Time Protocol (PTP) communication path, clock that is the source of time to which all other clocks on that path synchronize.

[SOURCE: IEC 61588:2009, 3.1.17]

3.6.49
member
piece of an attribute that is structured as an element of an array

3.6.50
Message Router
object within a node that distributes messaging requests to appropriate application objects

3.6.51
method
<object> synonym for an operational service which is provided by the server ASE and invoked by a client

3.6.52
module
hardware or logical component of a physical device

3.6.53
multipoint connection
connection from one node to many

Note 1 to entry: Multipoint connections allow messages from a single producer to be received by many consumer nodes.

3.6.54
network
set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.6.55
object
abstract representation of a particular component within a device, usually a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behavior

3.6.56
object specific service
service unique to the object class which defines it

3.6.57
ordinary clock
clock that has a single Precision Time Protocol (PTP) port in a domain and maintains the timescale used in the domain

Note 1 to entry: It may serve as a source of time, i.e., be a master clock, or may synchronize to another clock, i.e., be a slave clock.

[SOURCE: IEC 61588:2009, 3.1.22, modified – second sentence changed to a Note]

3.6.58
originator
client responsible for establishing a connection path to the target

3.6.59
parent clock
master clock to which a clock is synchronized

[SOURCE: IEC 61588:2009, 3.1.23]

3.6.60
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.6.61
physical device
automation or other network device

3.6.62
point-to-point connection
connection that exists between exactly two application objects

3.6.63
Precision Time Protocol
PTP
protocol defined by IEC 61588:2009

Note 1 to entry: As an adjective, it indicates that the modified noun is specified in or interpreted in the context of IEC 61588:2009.

[SOURCE: IEC 61588:2009, 3.1.28, modified – second sentence changed to a Note]

3.6.64
produce
act of sending data to be received by a consumer

3.6.65
producer
node that is responsible for sending data

3.6.66
property
general term for descriptive information about an object

3.6.67
PTP message
one of the message types defined in IEC 61588:2009

[SOURCE: IEC 61588:2009, 3.1.33]

3.6.68
PTP port
logical access point of a clock for PTP communications to the communications network

[SOURCE: IEC 61588:2009, 3.1.35]

3.6.69
resource
processing or information capability of a subsystem

3.6.70
serial number
unique 32-bit integer value assigned by each manufacturer/vendor to every device having Type 2 communication capabilities

Note 1 to entry: The Vendor ID and serial number jointly form a unique identifier for each device.

3.6.71
server
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

b) object which provides services to another (client) object

3.6.72
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.6.73
synchronized clocks
(to a specified uncertainty) two clocks which have the same epoch and for which measurements of the time of a single event at an arbitrary time differ by no more than the specified uncertainty

[SOURCE: IEC 61588:2009, 3.1.40, modified – reworded]

3.6.74
System Time
absolute time value as defined by CPF2 time synchronization in the context of a distributed time system where all devices have a local clock that is synchronized with a common master clock

Note 1 to entry: In the context of CPF2, System Time is a 64-bit integer value in units of nanoseconds with a value of 0 corresponding to the date 1970-01-01.

3.6.75
target
end-node to which a connection is established

3.6.76
transparent clock
device that measures the time taken for a Precision Time Protocol (PTP) event message to transit the device and provides this information to clocks receiving this PTP event message

[SOURCE: IEC 61588:2009, 3.1.46, modified – truncated]

3.6.77
Unconnected Message Manager
UCMM
component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object

3.6.78
unconnected service
messaging service which does not rely on the set up of a connection between devices before allowing information exchanges

3.6.79
vendor ID
identification of each product manufacturer/vendor by a unique number

Note 1 to entry: Vendor IDs are assigned by ODVA, Inc.

3.7 Type 2 abbreviated terms and symbols

AE Application Entity

AL Application layer

APO Application object

AP Application process

APDU Application protocol data unit

AR Application relationship

AREP Application relationship end point

ASCII American standard code for information interchange

ASE Application service element

CID Connection ID
CM_API Actual packet interval
CM_RPI Requested packet interval
Cnf Confirmation
CR Communication relationship
DL- (as a prefix) data-link-
DLC Data-link connection
DLCEP Data-link connection end point
DLL Data-link layer
DLM Data-link-management
DLSAP Data-link service access point
DLSDU DL-service-data-unit
FAL Fieldbus application layer
FIFO First-in first-out
ID Identifier
IEC International Electrotechnical Commission
Ind Indication
IP Internet protocol
ISO International Organization for Standardization
I/O Input/output
LME Layer management entity
O2T Originator to target (connection characteristics)
O⇒T Originator to target (connection characteristics)
OSI Open systems interconnect
PDU Protocol data unit
PL Physical layer
PTP Precision Time Protocol [IEC 61588:2009]
QoS Quality of service
REP Route endpoint
Req Request
Rsp Response
SAP Service access point
SDU Service data unit
SEM State event matrix
STD State transition diagram, used to describe object behavior
T2O Target to originator (connection characteristics)
T⇒O Target to originator (connection characteristics)

3.8 Conventions

3.8.1 Overview

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 two parts, its class specification, and its service specification.

The class specification defines the attributes of the class. The attributes are accessible from instances of the class using the Object Management ASE services specified in Clause 5 of this document. The service specification defines the services that are provided by the ASE.

3.8.2 General conventions

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

Bold font is used in this document to highlight parameter names or important requirement elements from surrounding text.

3.8.3 Conventions for class definitions

Class definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is shown below:

FAL ASE:			**ASE Name**
CLASS:			**Class name**
CLASS ID:			**#**
PARENT CLASS:			Parent class name
ATTRIBUTES:			
1	(o)	Key Attribute:	numeric identifier
2	(o)	Key Attribute:	name
3	(m)	Attribute:	attribute name(values)
4	(m)	Attribute:	attribute name(values)
4.1	(s)	Attribute:	attribute name(values)
4.2	(s)	Attribute:	attribute name(values)
4.3	(s)	Attribute:	attribute name(values)
5.	(c)	Constraint:	constraint expression
5.1	(m)	Attribute:	attribute name(values)
5.2	(o)	Attribute:	attribute name(values)
6	(m)	Attribute:	attribute name(values)
6.1	(s)	Attribute:	attribute name(values)
6.2	(s)	Attribute:	attribute name(values)
SERVICES:			
1	(o)	OpsService:	service name
2.	(c)	Constraint:	constraint expression
2.1	(o)	OpsService:	service name
3	(m)	MgtService:	service name

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

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

(3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the FAL ASE that will provide the services for this class. When qualified by the identity of its FAL ASE, it unambiguously identifies the class within the scope of the FAL. The value "NULL" indicates that the class cannot be instantiated. Class IDs between 1 and 99, 240 and 767 are reserved by this document to identify standardized classes. CLASS IDs between 100 and 199, 768 and 1 279 are allocated for identifying user defined classes.

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

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

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

a) Each of the attribute entries contains a line number in column 1, a mandatory (m) / optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label in column 3, a name or a conditional expression in column 4, and optionally a list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.
If an attribute is optional, its default value (specified in IEC 61158-6-2) shall provide the same behavior as if the attribute was not implemented.

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

c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting is used to specify

i) fields of a structured attribute (4.1, 4.2, 4.3),

ii) attributes conditional on a constraint statement (5). Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2).

iii) the selection fields of a choice type attribute (6.1 and 6.2).

(6) The "SERVICES" label indicates that the following entries are services defined for the class.

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

b) The label "OpsService" designates an operational service (1).

c) The label "MgtService" designates an management service (2).

d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

3.8.4 Conventions for service definitions

3.8.4.1 General

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

3.8.4.2 Service parameters

Service primitives are used to represent service user/service provider interactions (ISO/IEC 10731). They convey parameters which indicate information available in the user/provider interaction. In any particular interface, not all parameters need be explicitly stated.

The service specifications of this document uses a tabular format to describe the component parameters of the ASE service primitives. The parameters which apply to each group of service primitives are set out in tables. Each table consists of up to five columns for the

1) Parameter name,
2) request primitive,
3) indication primitive,
4) response primitive, and

5) confirm primitive.

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

M parameter is mandatory for the primitive

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

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

— (blank) parameter is never present.

S parameter is a selected item.

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

a) a parameter-specific constraint:

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

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

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

3.8.4.3 Service procedures

The procedures are defined in terms of

- the interactions between application entities through the exchange of fieldbus Application Protocol Data Units, and
- the interactions between an application layer service provider and an application layer service user in the same system through the invocation of application layer service primitives.

These procedures are applicable to instances of communication between systems which support time-constrained communications services within the fieldbus application layer.

4 Common concepts

The common concepts and templates used to describe the application layer service in this document are detailed in IEC 61158-1, Clause 9.

5 Data type ASE

5.1 General

An overview of the data type ASE and the relationships between data types is provided in IEC 61158-1, 10.2.

5.2 Formal definition of data type objects

The template used to describe the data type class in Clause 5 is detailed in IEC 61158-1, 10.2. This includes the specific ASE structure and the definition of its attributes.

5.3 FAL defined data types

5.3.1 Fixed length types

5.3.1.1 Boolean types

5.3.1.1.1 Boolean

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	1
2	Data type Name	=	Boolean
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This data type expresses a Boolean data type with the values TRUE and FALSE.

5.3.1.1.2 BOOL

This IEC 61131-3 type is the same as Boolean.

5.3.1.2 Bitstring types

5.3.1.2.1 BitString8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	22
2	Data type Name	=	Bitstring8
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	1

This type contains 1 element of type BitString.

5.3.1.2.2 SWORD

This IEC 61131-3 type is the same as Bitstring8.

5.3.1.2.3 BitString16

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	23
2	Data type Name	=	Bitstring16
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	2

5.3.1.2.4 WORD

This IEC 61131-3 type is the same as Bitstring16.

5.3.1.2.5 BitString32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	24
2	Data type Name	=	Bitstring32
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	4

5.3.1.2.6 DWORD

This IEC 61131-3 type is the same as Bitstring32.

5.3.1.2.7 BitString64

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	57
2	Data type Name	=	Bitstring64
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	8

5.3.1.2.8 LWORD

This IEC 61131-3 type is the same as Bitstring64.

5.3.1.3 Date types

5.3.1.3.1 DATE

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	DATE
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This IEC 61131-3 type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of two octets. It expresses the date as a number of days, starting from 1972-01-01 (January 1st, 1972), the start of the Coordinated Universal Time (UTC) era, until 2151-06-06 (June 6th, 2151), i.e. a total range of 65 536 days.

5.3.1.3.2 TimeOfDay

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date counting the days from 1984-01-01 (January 1st, 1984).

5.3.1.3.3 TimeOfDay without date indication

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	52
2	Data type Name	=	TimeOfDay without date indication
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type is composed of one element of an unsigned value and expresses the time of day. The element is an Unsigned32 data type and gives the time after the midnight in milliseconds.

5.3.1.3.4 TIME_OF_DAY

This IEC 61131-3 type is the same as TimeofDay without date indication.

5.3.1.4 Numeric types

5.3.1.4.1 Floating Point types

5.3.1.4.1.1 Float32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float32
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type has a length of four octets. The format for Float32 is that defined by ISO/IEC/IEEE 60559 as single precision.

5.3.1.4.1.2 REAL

This IEC 61131-3 type is the same as Float32.

5.3.1.4.1.3 Float64

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	15
2	Data type Name	=	Float64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This type has a length of eight octets. The format for Float64 is that defined by ISO/IEC/IEEE 60559 as double precision.

5.3.1.4.1.4 LREAL

This IEC 61131-3 type is the same as Float64.

5.3.1.4.2 Integer types

5.3.1.4.2.1 Integer8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2
2	Data type Name	=	Integer8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This integer type is a two's complement binary number with a length of one octet.

5.3.1.4.2.2 SINT

This IEC 61131-3 type is the same as Integer8.

5.3.1.4.2.3 Integer16

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	3
2	Data type Name	=	Integer16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is a two's complement binary number with a length of two octets.

5.3.1.4.2.4 INT

This IEC 61131-3 type is the same as Integer16.

5.3.1.4.2.5 Integer32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	4
2	Data type Name	=	Integer32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is a two's complement binary number with a length of four octets.

5.3.1.4.2.6 DINT

This IEC 61131-3 type is the same as Integer32.

5.3.1.4.2.7 Integer64

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	55
2	Data type Name	=	Integer64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This integer type is a two's complement binary number with a length of eight octets.

5.3.1.4.2.8 LINT

This IEC 61131-3 type is the same as Integer64.

5.3.1.4.3 Unsigned types

5.3.1.4.3.1 Unsigned8

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	5
2	Data type Name	=	Unsigned8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This type has a length of one octet.

5.3.1.4.3.2 USINT

This IEC 61131-3 type is the same as Unsigned8.

5.3.1.4.3.3 Unsigned16

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	6
2	Data type Name	=	Unsigned16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of two octets.

5.3.1.4.3.4 UINT

This IEC 61131-3 type is the same as Unsigned16.

5.3.1.4.3.5 Unsigned32

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	7
2	Data type Name	=	Unsigned32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of four octets.

5.3.1.4.3.6 UDINT

This IEC 61131-3 type is the same as Unsigned32.

5.3.1.4.3.7 Unsigned64

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	56
2	Data type Name	=	Unsigned64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This type is a binary number. The most significant bit of the most significant octet is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of eight octets.

5.3.1.4.3.8 ULINT

This IEC 61131-3 type is the same as Unsigned64.

5.3.1.5 Time types

5.3.1.5.1 TIME

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	not used
2	Data type Name	=	TIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This IEC 61131-3 type is a two's complement binary number with a length of four octets. The unit of time for this type is 1 ms.

5.3.1.5.2 ITIME

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	not used
2	Data type Name	=	ITIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This IEC 61131-3 type extension is a two's complement binary number with a length of two octets. The unit of time for this type is 1 ms.

5.3.1.5.3 FTIME

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	not used
2	Data type Name	=	FTIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

5.3.1.5.4 LTIME

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	not used
2	Data type Name	=	LTIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This IEC 61131-3 type extension is a two's complement binary number with a length of eight octets. The unit of time for this type is 1 µs.

5.3.2 String types

5.3.2.1 BitString

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	14
2	Data type Name	=	Bitstring
3	Format	=	STRING
5.1	Octet Length	=	1 to n

This string type is defined as a series of BitString8 elements.

5.3.2.2 OctetString

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	10
2	Data type Name	=	OctetString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet. IEC 61158-6-2 defines the order of transmission.

5.3.2.3 EPATH

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	EPATH
3	Format	=	STRING
4.1	Octet Length	=	1 to n

An EPATH is an ordered sequence of octets, numbered from 1 to n. Its format is further specified in IEC 61158-6-2, 4.1.9.

5.3.2.4 ETH_MAC_ADDR

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	ETH_MAC_ADDR
3	Format	=	ARRAY
6.1	Number of array elements	=	6
6.2	Array element data type	=	UINT

The Ethernet MAC Address is an array of 6 octets. The recommended display format is “XX-XX-XX-XX-XX-XX”, starting with the first octet. The Ethernet MAC Address is assigned by the manufacturer and shall be unique per ISO/IEC/IEEE 8802-3 requirements. The general requirement is that the value of this type shall be Ethernet MAC addresses used in Ethernet frames.

5.3.3 Structure types

5.3.3.1 DATE_AND_TIME

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	DATE_AND_TIME
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Time_Of_Day_Element
5.2.2	Field Data type	=	TIME_OF_DAY
5.3.1	Field Name	=	Date_Element
5.3.2	Field Data type	=	DATE

This IEC 61131-3 type extension is a structure which expresses both the date (as a number of days starting from 1972-01-01 until 2151-06-06),and the time of day as a number of ms starting from midnight.

5.3.3.2 SHORT_STRING

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	SHORT_STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data type	=	OctetString

This IEC 61131-3 type extension is composed of two elements. Charcount_Element gives the current number of characters in the Stringcontents_Element (one octet per character). Characters shall be as specified in ISO/IEC 8859-1.

5.3.3.3 STRING

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	UINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data type	=	OctetString

This IEC 61131-3 type is composed of two elements. Charcount_Element gives the current number of characters in the Stringcontents_Element (one USINT per character). Characters shall be as specified in ISO/IEC 8859-1.

5.3.3.4 STRING2

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	STRING2
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	UINT
5.3.1	Field Name	=	String2contents_Element
5.3.2	Field Data type	=	OctetString

This IEC 61131-3 data type extension is composed of two elements. Charcount_Element gives the current number of characters in the String2contents_Element (one UINT per character). Characters shall be as specified in ISO/IEC 10646.

5.3.3.5 STRINGN

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	STRINGN
3	Format	=	STRUCTURE
5.1	Number of Fields	=	3
5.2.1	Field Name	=	Charsize_Element
5.2.2	Field Data type	=	UINT
5.3.1	Field Name	=	Charcount_Element
5.3.2	Field Data type	=	UINT
5.4.1	Field Name	=	StringNcontents_Element
5.4.2	Field Data type	=	OctetString

This IEC 61131-3 type extension is composed of three elements. Charsize_Element gives the size of a character in StringNcontents_Element (N = number of USINT). Charcount_Element gives the current number of characters in the StringNcontents_Element (N USINT per character). Characters shall be as specified in ISO/IEC 10646.

5.3.3.6 STRINGI

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	STRINGI
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Stringnum_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	International_String_Array
5.3.2	Field Data type	=	STRINGI_ARRAY

This IEC 61131-3 type extension allows for multiple language representation for a single string. It is a structured data type which allocates a USINT variable (Stringnum_Element) containing the number of internationalized character strings and an array of structures (International_String_Array) containing the internationalized character strings.

The international character string structure (STRINGI_ELEMENT) is defined as follows:

- a USINT (Language1_Element) indicating the first ASCII character of the ISO 639-2/T language;
- a USINT (Language2_Element) indicating the second character of the ISO 639-2/T language;
- a USINT (Language3_Element) indicating the third character of the ISO 639-2/T language;
- a USINT (Datatype_Element), limited to the values 0xD0 (STRING), 0xD5 (STRING2), 0xD9 (STRINGN), and0xDA (SHORT_STRING) indicating the structure of the character string;
- a UINT (Charset_Element) indicating the character set which the character string is based on;
- an array of octet elements which is the actual international character string (which data type is specified in Datatype_Element).

The three characters for the language come from ISO 639-2/T (Alpha-3 Terminology Code), and the character set values come from IANA MIB printer codes (IETF RFC 1759). The character set values are limited to those values that are provided in Table 1.

Table 1 – Valid IANA MIB printer codes for character set selection

Character Set	Value
ISO 8859-1:1987	4
ISO 8859-2:1987	5
ISO 8859-3:1988	6
ISO 8859-4:1988	7
ISO 8859-5:1988	8
ISO 8859-6:1987	9
ISO 8859-7:1987	10
ISO 8859-8:1989	11
ISO 8859-9:1989	12
ISO/IEC 10646-UCS-2	1 000
ISO/IEC 10646-UCS-4	1 001

5.3.3.7 SHORT_STRING

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	not used
2	Data type Name	=	SHORT_STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data type	=	OctetString

This IEC 61131-3 type extension is composed of a single elements. Charcount_Element gives the current number of characters in the Stringcontents_Element (one octet per character). Characters shall be as specified in ISO/IEC 8859-1.

5.3.3.8 STRINGI_ELEMENT

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	not used
2	Data type Name	=	STRINGI_ELEMENT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	6
5.2.1	Field Name	=	Language1_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	Language2_Element
5.3.2	Field Data type	=	USINT
5.4.1	Field Name	=	Language3_Element
5.4.2	Field Data type	=	USINT
5.5.1	Field Name	=	Datatype_Element
5.5.2	Field Data type	=	EPATH
5.6.1	Field Name	=	Charset_Element
5.6.2	Field Data type	=	UINT
5.7.1	Field Name	=	Stringcontents _Element
5.7.2	Field Data type	=	SHORT_STRING | STRING | STRING2 | STRINGN

5.4 Data type ASE service specification

There are no operational services defined for the type object.

6 Communication model specification

6.1 Concepts

6.1.1 General

This is a serial communication system for communication between devices that wish to exchange both time critical and messaging type application information. These devices include simple I/O devices, such as sensors/actuators as well as complex control devices such as robots, programmable logic controllers, welders, process controllers.

Unlike some general purpose communication systems that rely on the destination delivery model, this network uses a variant of the publisher/subscriber push model, called the producer/consumer model. The producer/consumer model allows the exchange of time critical application information between a sending device (i.e. the producer) and many receiving devices (i.e. the consumers) without the need to send the data separately to each destination. This is accomplished by attaching a unique identifier to each piece of application information that is being produced onto the network medium. Any device that requires a specific piece of application information simply filters the data on the network medium for the appropriate identifier. Many devices can receive the same piece of application information from a single producing device.

The Type 2 application layer can be associated with different data-link layers, depending on the selected communication profile.

The Type 2 specific data-link layer provides a high degree of protocol efficiency by utilizing an implied token passing mechanism. This mechanism allows all devices equal access to the network without the network overhead associated with passing a “token” to each device granting it permission to send data. The protocol utilizes a time based scheduling mechanism which provides network devices with deterministic and predictable access to the medium while preventing network collisions. This scheduling mechanism allows time critical data, which is required on a periodic, repeatable and predictable basis, to be produced on a predefined schedule without the loss of efficiency associated with continuously requesting or “polling” for the required data. The protocol supports an additional mechanism which allows data that is not time critical in nature or which is only required on an occasional basis to utilize any available network time. This unscheduled data is transmitted after the production of the time critical data has been completed and before the beginning of the next scheduled production of time critical data.

6.1.2 General concepts

Most of the general concepts described in Clause 4 apply, with some restrictions or additions as noted:

- the application layer includes those functionalities from the intermediate layers which are necessary for proper mapping onto the data-link layer,
- Client/Server relationships can be one-to-one, but also one-to-many (see AR model in 6.3.1),
- a variant of the Publisher/Subscriber Push model, the Producer/Consumer model, is used as the base of all AR's (see AR model in 6.3.1),
- an overview of the ASE/APO types and their relationships is provided in 6.1.3,
- AR follow a model which is detailed in 6.3.1,
- specific naming and addressing is specified in 6.1.4,

- common FAL attributes and parameters listed in IEC 61158-1, 9.7 are not used; instead they are specified in relevant Type-specific subclauses,
- there are no Abort or Reject services, only negative result confirmations, so the error procedure described in Annex A does not apply.

6.1.3 Relationships between ASEs

Every node shall contain as a minimum instance one of the following ASE object classes in its application layer.

Object Management ASE:

- Identity (identification and general information about the device)
- Message Router (messaging connection point for communications within the device)

Connection Manager ASE:

- Connection Manager (establishment and maintenance of connections)

or Connection ASE:

- Connection (messaging connection point for communications within the device)

AR ASE:

- Unconnected Message Manager (queued messaging services on a single link, optional for CP 2/3)

Other application layer objects may be implemented in some nodes only:

Object Management ASE:

- Assembly object (binds data from multiple objects to be sent through one connection)
- Acknowledge Handler object (handles acknowledge messages from the application objects)
- Time Sync object (interface to the IEC 61588:2009 precision clock synchronization protocol)
- Parameter object (public interface to device configuration data)
- additional application-specific objects constructed according to the general Type 2 formal model

AR ASE:

- Transports (connected messaging and data services).

Yet more objects belong to system management, primarily concerned with the data-link layer:

- ControlNet object (station management interfaces to lower layers, required in all devices using Type 2 data-link layer),
- Keeper object (holds and distributes attributes of all devices on the link, one required per link, with optional backup(s) – used in conjunction with Type 2 data-link layer),
- Scheduling object (holds information on link schedules, one required in each connection originator – used in conjunction with Type 2 data-link layer),
- TCP/IP Interface object (provides the mechanism to configure the TCP/IP interface of a device, required in all devices with a TCP/IP interface),
- Ethernet Link object (maintains link-specific counters and status information for an Ethernet port, required in all devices with an Ethernet port),

- DeviceNet object (station management interfaces to lower layers, required in all devices using ISO 11898:1993 data-link layer),
- Connection Configuration object (interface to create, configure and control connections in a device),
- DLR object (configuration and status information interface for the DLR protocol),
- QoS object (configuration interface for QoS-related behaviors in devices),
- Port object (describes Type 2 ports present on the device),
- PRP/HSR Protocol object (configuration and diagnostic interface for the PRP and HSR protocols),
- PRP/HSR Nodes Table object (record of all PRP or HSR capable nodes detected on the network).

NOTE These lower layer management objects are described in IEC 61158-4-2.

The ASE and object interactions for a device using both application and data-link Type 2 protocols are illustrated in Figure 1:

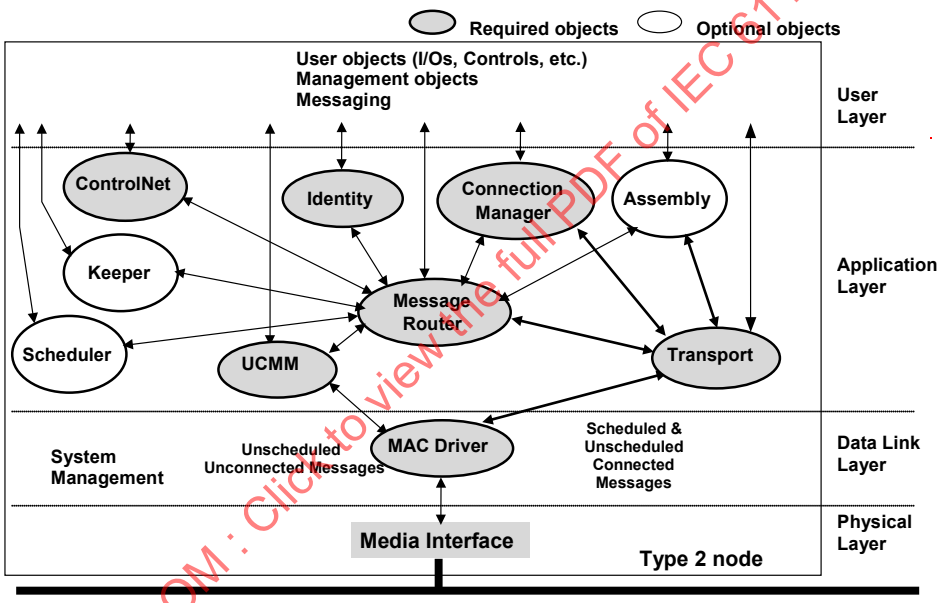


Figure 1 – Overview of ASEs and object classes

6.1.4 Naming and addressing

The information presented here provides a common basis for logically addressing separate physical components across the network. These addressing terms are used in the specifications. Figure 2 should be referenced in the following discussion.

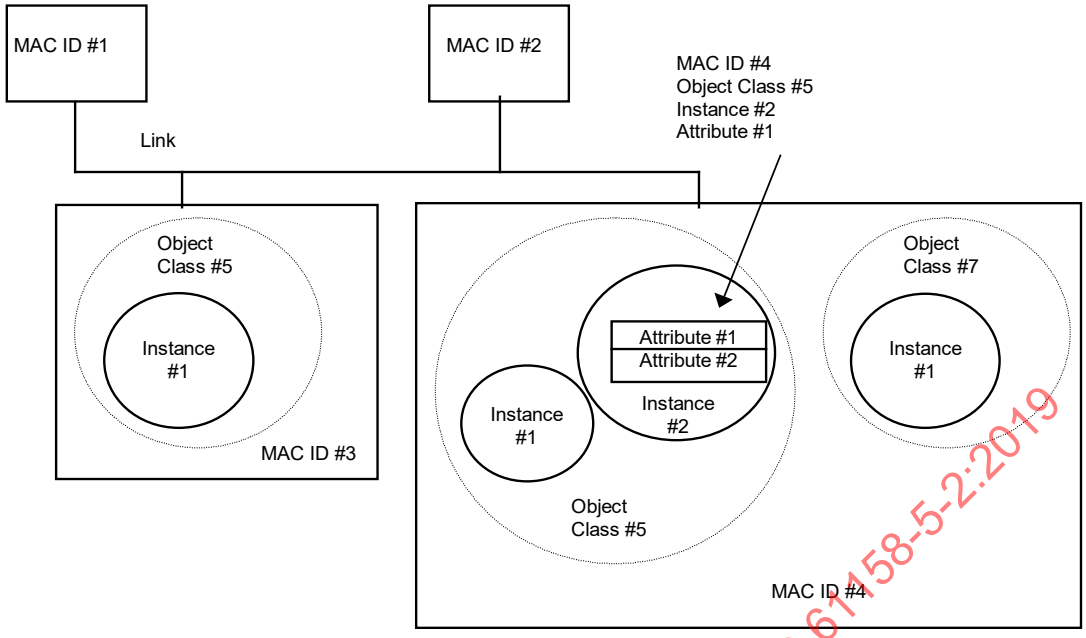


Figure 2 – Addressing format using MAC, class, instance and attribute IDs

The term “node” is used to refer to that portion of a device which contains a link interface and responds to a single MAC ID. The term “device” refers to a complete physical device connecting to the network. A device is able to contain multiple nodes.

NOTE When the Type 2 application layer is used in conjunction with Ethernet, the referenced MAC ID is actually the IP address and not the Ethernet MAC ID.

The Class ID is a unique integer identification value assigned to each object class accessible from the network. The object class may be referenced with this Class ID. In all IEC 61158 specifications for Type 2, class code is synonymous with Class ID.

The Instance ID is an integer identification value assigned to an object instance when it is created that identifies it among all Instances of the same Class. This integer is unique within the <Node:Class> in which it resides.

The Attribute ID is an integer identification value that is unique among all other attributes of that object.

The address of attribute number 1 of instance 2 in class 5 in the node with a MAC ID of 4 is MAC ID#4: Object Class #5: Instance #2: Attribute #1 as shown in Figure 2. This nomenclature is referred to as Class/Instance/Attribute addressing.

Information on how to address an object through multiple links or using a name is provided in IEC 61158-6-2 (Path definition).

6.1.5 Data types

6.1.5.1 Data type general specification

The specification of a data type is comprised of the range of values that variables of the type may take on and the operations performed on these variables.

Data is made up of elementary (primitive) data types. These elementary data types are used to construct derived (constructed) data types.

NOTE 1 Elementary and derived data type specifications correspond to the notation of IEC 61131-3. In addition, since function blocks as defined in IEC 61131-3 have both an associated data structure and a set of defined operations, these elements are also specified below as additional data types.

The derived data types are the following:

directly derived;

enumerated;

subrange;

structured;

array.

NOTE 2 These derived data types are defined in IEC 61131-3:2003, 1.3 and 2.3.3. The means of specifying these data types and their default initial values is defined in IEC 61131-3:2003, 2.3.3.1 and 2.3.3.2. The usage of variables of these data types is defined in IEC 61131-3:2003, 2.3.3.3.

6.1.5.2 Supported elementary data types

The following basic data types defined in Clause 5 are supported:

BOOL
SINT
INT
DINT
LINT
USINT
UINT
UDINT
ULINT
REAL
LREAL
ITIME
TIME
FTIME
LTIME
DATE
TIME_OF_DAY or TOD
DATE_AND_TIME or DT
STRING
STRING2
STRINGN
STRINGI
SHORT_STRING
SWORD
WORD
DWORD
LWORD
EPATH

6.1.5.3 Compliance with IEC 61131-3

6.1.5.3.1 Compliance statement

NOTE 1 Subclause 6.1.5.3 provides information with respect to only the data types and associated operations defined in this part of IEC 61158.

NOTE 2 IEC 61131-3:2003, 1.5.1, defines the requirements which are met by programmable controller systems claiming compliance to IEC 61131-3. This provides the data type-related information to be included in the documentation of functional units which support the data types defined in this part of IEC 61158. Subsets or extensions of this documentation are provided as appropriate to the specific compliant functional unit.

An implementation shall comply with the requirements of IEC 61131-3 for language features as specified in Table 2 through Table 4.

Table 2 – Common elements

IEC 61131-3:2003 Table/feature	Feature description
10/1	BOOL data type
10/2	SINT data type
10/3	INT data type
10/4	DINT data type
10/5	LINT data type
10/6	USINT data type
10/7	UINT data type
10/8	UDINT data type
10/9	ULINT data type
10/10	REAL data type
10/11	LREAL data type
10/12	TIME data type
10/13	DATE data type
10/14	TIME_OF_DAY or TOD data type
10/15	DATE_AND_TIME or DT data type
10/16	STRING data type
10/17	SWORD data type
10/18	WORD data type
10/19	DWORD data type
10/20	LWORD data type
12/1	Directly derived data types
12/2	Enumerated data types
12/3	Subrange data types
12/4	Array data types
12/5	Structured data types
13	Standard default initial values
Subclause 2.5.1.3	User-declared functions (no table entry)
22/1	Type conversions (see Table 4)
22/2	TRUNC function
22/3	BCD_TO_** functions
22/4	*_TO_BCD functions
23/1-11	Standard functions of one numeric variable: ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN

IEC 61131-3:2003 Table/feature	Feature description
24/12n-18n	Standard named arithmetic functions: ADD, MUL, SUB, DIV, MOD, EXPT, MOVE
24/ 12s-15s, 17s,18s	Standard symbolic arithmetic functions: +, *, –, /, **, :=
25/1-4	Standard bit string functions: SHL, SHR, ROR, ROL
26/5s-8s	Standard named bitwise Boolean functions: AND, OR, XOR, NOT
26/5s-7s	Standard symbolic bitwise Boolean functions: &, >=1, =2k+1
27/1-4	Standard selection functions: SEL, MAX, MIN, LIMIT, MUX
28/5n-10n	Standard named comparison functions: GT, GE, EQ, LE, LT, NE
28/5s-10s	Standard symbolic comparison functions: >, >=, =, <=, <, <>
29/1-9	Standard character string functions: LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND
30/1-14	Standard functions of time data types: (see NOTE) ADD, SUB, MUL, DIV, CONCAT, DATE_AND_TIME_TO_TIME_OF_DAY, TIME_OF_DAY_TO_DATE_AND_TIME
31/1-4	Standard functions of enumerated data types: SEL, MUX, EQ, NE
32	Standard access mechanisms to function block I/O parameters
33/8a,8b,9a,9b	User-defined function blocks per IEC 61131-3:2003, 2.5.2.2, with graphical or textual rising or falling edge input options
34/1-3	Standard bistable function blocks: SR, RS, SEMA
35/1,2	Standard edge detection function blocks: R_EDGE, F_EDGE
36/1-3	Standard counter function blocks: CTU, CTD, CTUD
37/1,2a,3a, 4	Standard timer function blocks: TP, TON, TOF, RTC
55/1-17	Standard operators: (), function evaluation, **,–, NOT, *, /, MOD, +,–, <, >, <=, >=, =, <>, &, AND, XOR, OR
NOTE IEC 61131-3:2003, Table 30, limits the data types to which these operations apply.	

Table 3 – ST language elements

IEC 61131-3:2003 Table/feature	Feature description
55/1-17	Standard operators: (), function evaluation, **,–, NOT, *, /, MOD, +,–, <, >, <=, >=, =, <>, &, AND, XOR, OR
56/2	Function block invocation and output usage

Table 4 – Type conversion operations

Operation	Result	Error conditions
ANY_BIT_TO_ANY_BIT	See note 4	None
ANY_BIT_TO_ANY_INT	OUTmin + Sbk2k (note 5)	Result > OUTmax
ANY_BIT_TO_BOOL	FALSE if IN = 0 TRUE otherwise	None
ANY_BIT_TO_STRING	See note 6	None
ANY_DATE_TO_STRING	See note 7	None
ANY_INT_TO_BOOL	FALSE if IN = 0 TRUE otherwise	None
ANY_NUM_TO_ANY_INT	IN (note 8)	(IN > OUTmax) or (IN < OUTmin)
ANY_NUM_TO_ANY_REAL	IN	See note 9
ANY_NUM_TO_DATE	See note 10	
ANY_NUM_TO_STRING	See note 11	
ANY_NUM_TO_TIME	See note 12	
ANY_NUM_TO_TOD	See note 13	
ANY_REAL_TO_BOOL	FALSE if IN = 0,0 TRUE otherwise	None
BOOL_TO_ANY_BIT BOOL_TO_ANY_INT	0 if IN = FALSE 1 if IN = TRUE	None
BOOL_TO_ANY_REAL	0,0 if IN = FALSE 1,0 if IN = TRUE	None
BOOL_TO_STRING	'FALSE' if IN = FALSE 'TRUE' if IN = TRUE	None
DATE_TO_ANY_NUM	See note 14	Result > OUTmax
STRING_TO_ANY	Converted data	See note 15
TIME_TO_ANY_NUM	See note 16	Result > OUTmax
TOD_TO_ANY_NUM	See note 17	Result > OUTmax

NOTE 1 Use of the generic data types ANY_NUM, ANY_REAL, ANY_INT, and ANY_BIT defined in IEC 61131-3:2003, 2.3.2, is intended to imply a family of conversions. For instance, the conversion BOOL_TO_ANY_REAL is intended to imply BOOL_TO_REAL and BOOL_TO_LREAL.

NOTE 2 IN refers to the value of the input variable of the type conversion function.

NOTE 3 OUT_{min} and OUT_{max} refer to the minimum and maximum values of the output data type of the conversion function, as defined in Clause 5.

NOTE 4 In conversions of bit string types, if the number of bits in the input variable IN is less than the number of the bits in the output variable OUT, the bits of the input is copied to the corresponding least significant bits of the result and the remainder of the result is zero-filled. If the number of bits of IN is greater than the number of bits of OUT, the least significant bits of IN is copied to the corresponding bits of the result. For instance:

SWORD_TO_WORD(16#FF) = 16#00FF
and
WORD_TO_SWORD (16#0FF0) = 16#F0

NOTE 5 Bit numbering in this expression is as specified in IEC 61158-6-2.

NOTE 6 The result of conversion of a bit string variable to type STRING consists of a string containing the base 16 literal representation of the variable value, as defined in IEC 61131-3:2003, 2.2.1, in characters taken from the ISO/IEC 646 character set.

NOTE 7 The result of conversion of a date and/or time of day variable to type STRING consists of a string containing the literal representation of the variable value, as defined in IEC 61131-3:2003, 2.2.1, in characters taken from the ISO/IEC 646 character set.

NOTE 8 Conversion of REAL and LREAL to integer types is accomplished by rounding as specified in ISO/IEC/IEEE 60559.

NOTE 9 Rounding errors can occur if the number of significant bits in the input variable is larger than the number of significant bits in the output floating-point representation. Also, range errors of the type noted for ANY_NUM_TO_ANY_INT can occur in LREAL_TO_REAL.

NOTE 10 Conversion of a variable of a numeric type to DATE has the same result as conversion of the variable to UINT, with the result being interpreted as the number of days since 1972-01-01.

NOTE 11 Conversion of a variable of a numeric type to type STRING consists of a string containing the literal representation of the variable value, as defined in IEC 61131-3:2003, 2.2.1, in characters taken from the ISO/IEC 646 character set.

NOTE 12 Conversion of a variable of a numeric type to TIME has the same result as conversion of the variable to DINT, with the result being interpreted as a duration in milliseconds.

NOTE 13 Conversion of a variable of a numeric type to TOD has the same result as conversion of the variable to UDINT, with the result being interpreted as a time since midnight in milliseconds.

NOTE 14 Conversion of a variable of type DATE to a numerical type is the same as the conversion of a variable of type UINT to the corresponding numerical type, with the result being the numerical equivalent of the days since 1972-01-01.

NOTE 15 It is an error if the STRING data to be converted is not in the format for external representation of the output data type as specified in IEC 61131-3:2003, 2.2, or if the result of the conversion is outside the range $\{OUT_{min}..OUT_{max}\}$.

NOTE 16 Conversion of a variable of type TIME to a numerical type is the same as the conversion of a variable of type DINT to the corresponding numerical type, with the result being the numerical equivalent of the corresponding time interval expressed in milliseconds.

NOTE 17 Conversion of a variable of type TOD (TIME_OF_DAY) to a numerical type is the same as the conversion of a variable of type UDINT to the corresponding numerical type, with the result being the numerical equivalent of the time since midnight expressed in milliseconds.

6.1.5.3.2 Implementation dependant parameters

The values of implementation dependent parameters from IEC 61131-3:2003, Table D.1, are as shown in Table 5.

NOTE Values of other implementation dependent parameters are defined in other standards or in the specifications of individual functional units as appropriate.

Table 5 – Values of implementation-dependent parameters

Subclause of IEC 61131-3:2003	Parameter	Value
2.2.3.1	Range of values of duration	Same as LINT in microseconds
2.3.1	Range of values for variables of type TIME	Same as DINT in milliseconds
	Precision of representation of seconds in types TIME_OF_DAY and DATE_AND_TIME	1 ms
2.3.3.1	Maximum number of enumerated values	256
2.3.3.2	Default maximum length of STRING variables	256
	Maximum allowed length of STRING variables	65 536
2.4.1.2	Maximum number of subscripts	8
	Maximum range of subscript values	0 – 255
	Maximum number of levels of structures	8
2.5.1.5	Maximum inputs of extensible functions	8
2.5.1.5.1	Effects of type conversions on accuracy	As defined in Table 4
2.5.1.5.2	Accuracy of functions of one variable	As defined in ISO/IEC/IEEE 60559
2.5.2.3.3	PVmin, PVmax of counters	0, 65 535

6.1.5.3.3 Language extensions

The extensions to IEC 61131-3 defined in this part are listed in Table 6. When these extensions are used in a particular device, the subclause references in this table shall be replaced by references to the descriptions of the corresponding extensions in the functional unit's documentation.

Table 6 – Extensions to IEC 61131-3:2003

Subclause	Description
2.1.1	ITIME data type FTIME data type LTIME data type STRING2 data type STRINGN data type SHORT_STRING data type STRINGI data type EPATH data type
2.1.4	Structured bit string types
2.2	Operations on STRING2 variables
2.2.4.1	Numbered bit string access
2.2.4.2	Structured bit string access

6.2 ASEs

6.2.1 Object management ASE

6.2.1.1 Overview

The FAL Object Management ASE is used to access all network accessible application elements or system management elements.

An access rule is associated with each attribute of an object class, which specifies how a requester can access this attribute. The definitions for access rules are as follows:

- Settable (Set) – The attribute shall be accessed by at least one of the set services (for enumerated attributes, if the device supports only one of the values and the object definition does not require more than one value to be supported, then the attribute may be implemented as Get only);

  Settable attributes, unless otherwise specified by the object definition, shall be also Gettable and may be accessed by get services;
- Gettable (Get) – The attribute shall be accessed by at least one of the get services.

Some attributes may require non-volatile storage in order for their values to be maintained through power cycles. When relevant in object definitions, this is specified as follows:

- non-volatile (NV) indicates that value shall be saved;
- volatile (V) indicates that value shall not be saved.

6.2.1.2 FAL management model class specification

6.2.1.2.1 General formal model

6.2.1.2.1.1 Class definition

The base AP class below specifies the common attributes and services defined for all other AP classes, including those defined by the user (application specific).

(*) in front of an attribute or a service means that this attribute/service is either mandatory or optional, based on some constraints defined in the attribute/service description.

FAL ASE:			**FAL Management ASE**
CLASS:			**Base_Object**
CLASS ID:			**NULL**
PARENT CLASS:			**TOP**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(*)	Attribute:	Revision
2	(o)	Attribute:	Max Instance
3	(o)	Attribute:	Number of Instances
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):

1	(o)	Attribute:	Attr1

SYSTEM MANAGEMENT SERVICES:

1	(o)	Mgt Service:	Get_Attributes_All
2	(o)	Mgt Service:	Set_Attributes_All
3	(o)	Mgt Service:	Get_Attribute_List
4	(o)	Mgt Service:	Set_Attribute_List
5	(o)	Mgt Service:	Reset
6	(o)	Mgt Service:	Start
7	(o)	Mgt Service:	Stop
8	(o)	Mgt Service:	Create
9	(o)	Mgt Service:	Delete
13	(o)	Mgt Service:	Apply_Attributes
14	(o)	Mgt Service:	Get_Attribute_Single
16	(o)	Mgt Service:	Set_Attribute_Single
17	(o)	Mgt Service:	Find_Next_Object_Instance
21	(o)	Mgt Service:	Restore
22	(o)	Mgt Service:	Save
23	(o)	Mgt Service:	NOP
24	(o)	Mgt Service:	Get_Member
25	(o)	Mgt Service:	Set_Member
26	(o)	Mgt Service:	Insert_Member
27	(o)	Mgt Service:	Remove_Member
28	(o)	Mgt Service:	Group_Sync

OBJECT MANAGEMENT SERVICES:

1	(o)	Ops Service:	Get_Attributes_All
2	(o)	Ops Service:	Set_Attributes_All
3	(o)	Ops Service:	Get_Attribute_List
4	(o)	Ops Service:	Set_Attribute_List
5	(o)	Ops Service:	Reset
6	(o)	Ops Service:	Start
7	(o)	Ops Service:	Stop
8	(o)	Ops Service:	Create
9	(o)	Ops Service:	Delete
10	(o)	Ops Service:	Multiple_Service_Packet
13	(o)	Ops Service:	Apply_Attributes
14	(o)	Ops Service:	Get_Attribute_Single
16	(o)	Ops Service:	Set_Attribute_Single
21	(o)	Ops Service:	Restore
22	(o)	Ops Service:	Save
23	(o)	Ops Service:	NOP
24	(o)	Ops Service:	Get_Member
25	(o)	Ops Service:	Set_Member
26	(o)	Ops Service:	Insert_Member
27	(o)	Ops Service:	Remove_Member
28	(o)	Ops Service:	Group_Sync

OBJECT SPECIFIC SERVICES:

1	(o)	Mgt/Ops Service:	Service1

6.2.1.2.1.2 Access attributes

These internal attributes uniquely identify an object instance within a device. The corresponding values are used as part of the path in service requests to specify the target object instance.

Object Instance number

Unique number associated with a given object instance. Instance number 0 (zero) means that access to the object class itself is requested.

Symbolic name

Optional name which may be associated with a given object instance.

6.2.1.2.1.3 System management attributes (class attributes)

Attributes at the class level. An attribute whose value is shared by all objects within the same class is referred to as a Class Attribute.

Seven predefined Class Attribute IDs are reserved for class object definition. The seven predefined/reserved class attributes have the definitions listed below. Because these attributes are reserved, class Attribute ID numbers 1 through 7 are always reserved. Therefore, if a class attribute is added to an object specification, it shall start with AttributeID #8.

If a Class Attribute is optional, then a default value or a special case processing method shall be defined such that the Client (Requester) can process the error message that occurs when accessing those objects that choose not to implement the class attribute.

All the common class attributes have an access rule of Get only.

Revision

Revision of this object. The starting value assigned to this attribute is one (1). If updates that require an increase in this value are made, then the value of this attribute increases by 1.

If the value is 1, then this attribute is optional in implementations. If the value is greater than 1, then this attribute is mandatory. The Revision of an object specifies the interface to that object, which shall encompass all of the items in the object specification, including services, attributes, connections and behavior.

Max Instance

Maximum instance number of an object currently created in this class level in the device. The largest instance number of an object at this class hierarchy level created in the device.

Number of Instances

Number of object instances currently created at this class level in the device. The number of object instances at this class hierarchy level.

Optional attribute list

List of optional instance attributes utilized in an object class implementation. A list of attribute numbers specifying the optional attributes implemented in the device for this class.

Number of attributes

Number of attributes in the optional attribute list.

Optional attributes

List of optional attribute numbers.

Optional service list

List of optional services utilized in an object class implementation. A list of service codes specifying the optional services implemented in the device for this class.

If an optional service is implemented in a class, and the Optional Service List class attribute is also implemented in the class, then the service shall be included in the Optional Service List.

Number services

Number of services in the optional service list.

Optional services

List of optional service codes.

Maximum ID Number Class Attributes

The Attribute ID number of the last class attribute of the class definition implemented in the device.

NOTE This allows to simplify auto determination of class implementation by a remote terminal.

Maximum ID Number Instance Attributes

The Attribute ID number of the last instance attribute of the class definition implemented in the device.

NOTE This allows to simplify auto determination of class implementation by a remote terminal.

6.2.1.2.1.4 Object management attributes

Attributes at the instance level.

An Instance Attribute is an attribute whose value is unique to an object instance and whose definition is shared by all instances of an object. Each instance need only support the optional attributes that apply to it. If an instance does not support an optional attribute, the Attribute Not Supported (General Status code 0x14) error shall be returned for services targeting that attribute.

Instance Attributes are defined in the same terms as Class Attributes. There are no reserved Instance Attributes.

Instance ID = 0 is a special case. A service directed to Instance ID = 0 shall be applied to the class, not to a particular instance.

Each instance has a unique set of attributes. Instance attribute ID's may be numbered from 1 onwards without any correlation to the Attribute ID's of the class of which the object is an instance. There are no reserved Instance Attributes.

6.2.1.2.1.5 System management (class level) and object management (instance level)services

All the common services behave the same way, whether used as System Management or Object Management services.

NOTE 1 Class level is invoked by specifying an object instance ID of zero (see IEC 61158-6-2, 4.1.9.4.1).

Get_Attributes_All

The Get_Attributes_All service returns the contents of all attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

Set_Attributes_All

The Set_Attributes_All service modifies the contents of all attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

Get_Attribute_List

The Get_Attribute_List service returns the contents of the selected gettable attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

Set_Attribute_List

The Set_Attribute_List service updates the contents of the selected attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

Reset

The Reset service invokes the Reset service of the specified APO object class or instance.

NOTE 2 Typically this would cause a transition to a default state or mode.

Start

The Start service invokes the Start service of the specified APO object class or instance.

NOTE 3 Typically this would place an object instance into a running state or mode.

Stop

The Stop service invokes the Stop service of the specified APO object class or instance.

NOTE 4 Typically this would place an object instance into a stopped or idle state or mode.

Create

The Create service results in the instantiation of a new object instance within the specified object class.

Delete

The Delete service deletes an object instance of the specified object class.

Multiple_Service_Packet

The Multiple_Service_Packet service performs a set of services as an autonomous sequence.

Apply_Attributes

The Apply_Attributes service causes attribute values whose use is pending to become actively used in the specified APO object class or instance.

Get_Attribute_Single

The Get_Attribute_Single service returns the contents of the specified attribute of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

Set_Attribute_Single

The Set_Attribute_Single service modifies the contents of the specified attribute of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

Find_Next_Object_Instance

The Find_Next_Object_Instance service shall be supported at the APO object class level only. It causes the specified APO object class to search for and return a list of Instance IDs associated with existing object instances. Existing objects are those that are currently accessible from the link.

Restore

The Restore service restores the contents of an APO object class or instance attributes from a storage location accessible by the Save service. Attribute data is copied from a storage area to the currently active memory area used by the object class/instance.

Save

The Save service copies the contents of an APO object class or instance attributes to a location accessible by the Restore service.

NOP

The NOP service causes the receiving APO object instance to return a *No Operation* response, without carrying out any other internal action.

NOTE 5 This service can be used to test whether or not a particular object class/instance is still present and responding without causing a state change.

Get_Member

The Get_Member service returns member(s) information from within an attribute.

Set_Member

The Set_Member service sets member(s) information in an attribute.

Insert_Member

The Insert_Member service inserts member(s) into an attribute.

Remove_Member

The Remove_Member service removes member(s) from an attribute.

Group_Sync

The Group_Sync service verifies that each member of a group is synchronized to System Time.

6.2.1.2.1.6 Object specific services

Object-specific services are unique services supported only by the class of objects in which they are defined, whereas common services may be used in many objects. Object-specific service codes shall be unique only within the class in which they are defined.

Object-specific services sent to the class level of an object are called object-specific system management (class level) services. Object-specific services sent to the instance level of an object are called object-specific object management (instance level) services.

Class level is required by specifying an object instance ID of zero (see IEC 61158-6-2, 4.1.9.4.1).

6.2.1.2.2 Identity formal model

6.2.1.2.2.1 Class definition

The Identity ASE provides identification of and general information about the device and optionally its subsystems. Instance one (1) of the Identity object shall be present in all devices.

Instance one (1) shall identify the whole device. It alone shall be used for electronic keying, by applications wishing to determine what nodes are on the network and to match an EDS with a product on the network.

Other instances are optional. They may be provided by a device to give additional information about a device, such as its embedded components and/or subsystems.

Instances greater than one (1) that are used to report the revision of embedded components shall be assigned the Embedded Component device type. These instances exist to enable the manufacturer to report the revision of the embedded components of its choosing.

Instances greater than one (1) may also be used to identify internal subsystems that execute semi-independently (for example, a device may be designed to accept a third party communications card).

(*) in front of an attribute or a service means that this attribute/service is either mandatory or optional, based on some constraints defined in the attribute/service description.

FAL ASE:			**FAL Management ASE**
CLASS:			**Identity_Object**
CLASS ID:			**1**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
2	(*)	Attribute:	Max Instance
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(m)	Attribute:	Vendor ID
2	(m)	Attribute:	Device Type
3	(m)	Attribute:	Product Code
4	(m)	Attribute:	Revision
4.1	(m)	Attribute:	Major Revision
4.2	(m)	Attribute:	Minor Revision
5	(m)	Attribute:	Status
5.1	(m)	Attribute:	Owned
5.2	(m)	Attribute:	Configured
5.3	(m)	Attribute:	Extended Device Status
5.4	(m)	Attribute:	Minor Recoverable Fault
5.5	(m)	Attribute:	Minor Unrecoverable Fault
5.6	(m)	Attribute:	Major Recoverable Fault

5.7	(m)	Attribute:	Major Unrecoverable Fault
5.8	(m)	Attribute:	Extended Device Status 2
6	(m)	Attribute:	Serial Number
7	(m)	Attribute:	Product Name
8	(o)	Attribute:	State
9	(o)	Attribute:	Configuration Consistency Value
10	(o)	Attribute:	Heartbeat Interval
11	(o)	Attribute:	Active Language
12	(o)	Attribute:	Supported Language List
13	(o)	Attribute:	International Product Name
14	(o)	Attribute:	Semaphore
14.1	(m)	Attribute:	Client Electronic Key
14.2	(m)	Attribute:	Semaphore Timer
15	(o)	Attribute:	Assigned Name
16	(o)	Attribute:	Assigned Description
17	(o)	Attribute:	Geographic Location
18	(*)	Attribute	Type15 Identity Info
19	(o)	Attribute	Protection Mode
20	(o)	Attribute	Uptime
SYSTEM MANAGEMENT SERVICES:			
1	(o)	Mgt Service:	Get_Attributes_All
5	(o)	Mgt Service:	Reset
14	(*)	Mgt Service:	Get_Attribute_Single
17	(*)	Mgt Service:	Find_Next_Object_Instance
24	(o)	Mgt Service:	Get_Member
OBJECT MANAGEMENT SERVICES:			
1	(*)	Ops Service:	Get_Attributes_All
5	(m)	Ops Service:	Reset
14	(m)	Ops Service:	Get_Attribute_Single
16	(*)	Ops Service:	Set_Attribute_Single
24	(*)	Ops Service:	Get_Member
OBJECT SPECIFIC SERVICES:			
75	(o)	Ops Service:	Flash_LEDs

6.2.1.2.2.2 System management attributes (class attributes)

Max instance

If multiple instances of the Identity object exist, then this attribute is **mandatory**, else it is **optional**.

6.2.1.2.2.3 Object management attributes

The following instance attributes shall be used to identify with certainty the appropriate device targeted by a connection originator:

1) Vendor ID;
2) Device Type;
3) Product Code;
4) Revision.

This collection of attributes, when kept by the connection originator, is referred to as a device's "electronic key".

Vendor ID

Identification of each manufacturer/vendor by number. Vendor IDs uniquely identify a particular manufacturer/vendor. The value zero shall not be used.

This non-volatile instance attribute has an access rule of Get only.

Device type

Indication of general type of product. In order to allow interoperability and interchangeability, a Device Type shall be used to identify similar devices which exhibit the same behavior, produce and/or consume the same set of data, and contain the same set of configurable attributes. The formal definition of this information is known as a Device Profile: all devices with the same Device Type number shall meet the minimum requirements and implement the common options specified in the Device Profile for that device type. A listing of the Device Type ranges can be found in IEC 61158-6-2. The Embedded Component device type is not allowed for instance 1.

This non-volatile instance attribute has an access rule of Get only.

Product code

Identification of a particular product of an individual manufacturer. The manufacturer assigned Product Code identifies a particular product within a device type. Each manufacturer shall assign this code to each of its products. The Product Code typically maps to one or more catalogue/model numbers. Products shall have different codes if their configuration and/or runtime options are different. Such devices present a different logical view to the network. The value zero is not valid.

This non-volatile instance attribute has an access rule of Get only.

Revision

Revision of the item that this instance of the Identity object represents. The Revision attribute, which consists of Major and Minor Revisions, identifies the Revision of the item the Identity object is representing.

The value zero is not valid for either the Major or Minor Revision fields of a compliant product. If the Device Type of the instance is Embedded Component), the Major Revision and/or Minor Revision may be zero.

NOTE 1 The Major and Minor Revision are typically displayed as "major.minor", minor revisions being displayed as three digits with leading zeros as necessary.

This non-volatile instance attribute has an access rule of Get only.

Major revision

The major revision should be incremented by the manufacturer when there is a significant change to the ‘fit, form, or function’ of the product. Any changes that affect the configuration choices available to the user (and therefore the Electronic Data Sheet) require incrementing the major revision.

Minor revision

The minor revision is typically used to identify changes in a product that do not affect user configuration choices, e.g. bug fixes, hardware component change, labeling change. Changes in minor revision are not used by a configuration tool to match a device with an Electronic Data Sheet.

Status

Summary status of device. This attribute represents the current status of the entire device. Its value changes as the state of the device changes. The detailed format of this Status attribute is described in IEC 61158-6-2.

NOTE 2 The events that constitute a fault (recoverable or unrecoverable) are determined by the device implementers.

This volatile instance attribute has an access rule of Get only.

Owned

A (TRUE) indicates the device (or an object within the device) has an owner. Within the master/slave paradigm the setting of this bit means that the Predefined Master/Slave Connection Set has been allocated to a master. Outside the master/slave paradigm the meaning of this bit is to be defined.

Configured

A (TRUE) indicates the application of the device has been configured to do something different than the "out–of–box" default. This shall not include configuration of the communications.

Extended device status

This attribute is used to provide more detailed information about device status. Its values are either vendor specific, or as specified in IEC 61158-6-2, 4.1.8.2.1.2. The EDS shall indicate if the device instance (1) follows a vendor-specific definition for these bits. No mechanism is provided to enumerate the vendor-specific use of these values in the EDS file for instances greater than one (1).

Minor recoverable fault

A (TRUE) indicates the device detected a problem with itself, which is thought to be recoverable. The problem shall not cause the device to go into one of the faulted states.

NOTE 3 For example, an analogue input device is sensing an input that exceeds the configured maximum input value.

Minor unrecoverable fault

A (TRUE) indicates the device detected a problem with itself, which is thought to be unrecoverable. The problem shall not cause the device to go into one of the faulted states.

NOTE 4 For example, the device's battery backed RAM requires a battery replacement. The device is required to continue functioning properly until the first time power is cycled.

Major recoverable fault

A (TRUE) indicates the device detected a problem with itself, which caused the device to go into the "Major Recoverable Fault" state.

NOTE 5 For example, the device's configuration is incorrect or incomplete.

Major unrecoverable fault

A (TRUE) indicates the device detected a problem with itself, which caused the device to go into the "Major Unrecoverable Fault" state.

A device may not be able to communicate in the Major Unrecoverable Fault state. Therefore, it might not be able to report a Major Unrecoverable Fault. It shall not process a Reset service. The only exit from a Major Unrecoverable Fault shall be to cycle the device power.

NOTE 6 For example, the device failed its ROM checksum process.

Extended device status 2

This attribute is used to provide more detailed information about device status. Its values are vendor specific. The EDS shall indicate if the device instance (1) follows a vendor-specific definition for these bits. No mechanism is provided to enumerate the vendor-specific use of these values in the EDS file for instances greater than one (1).

Serial number

Serial number of device. This attribute provides a number used in conjunction with the Vendor ID to form a unique identifier for each device on any Type 2 network. Each manufacturer is responsible for guaranteeing the uniqueness of the serial number across all of its devices.

This non-volatile instance attribute has an access rule of Get only.

Product name

Human readable identification. This text string shall represent a short description of the product represented by the "Product Code" attribute. The same product code may have a variety of product name strings.

NOTE 7 The logical view to the network (see description of Product code above) does not include the Product name.

This non-volatile instance attribute has an access rule of Get only.

State

Indication of the present state of the device, as represented by the state transition diagram.

NOTE 8 The nature of a Major Unrecoverable Fault could be such that it could be not accurately reflected by the State attribute.

This volatile instance attribute has an access rule of Get only.

Configuration consistency value

Contents identify configuration of device. Indicates that a change in configuration has occurred.

A product may automatically modify the Configuration Consistency Value whenever any non-volatile attribute is altered. A client node may, or may not, compare this value to a value within its own memory prior to system operation.

NOTE 9 The client node's behavior, upon detection of a mismatch, is vendor specific. The Configuration Consistency Value could be a CRC, incrementing count or any other mechanism.

The only requirement is that if the configuration changes, the Configuration Consistency Value should be different to reflect the change.

This non-volatile instance attribute has an access rule of Get only.

Heartbeat interval

Sets the nominal interval between production of optional heartbeat messages.

This non-volatile instance attribute has an access rule of Get/Set.

Active language

Currently active language for the device. If this attribute is supported, all character strings within the device of data type STRING or SHORT_STRING shall follow this setting. Only languages supported by the ISO/IEC 8859-1 character set can be represented in these

strings; if the Active Language is set to a language which cannot be represented in ISO/IEC 8859-1, the device shall return the string in the default language. Any other object attributes (class or instance level) which set the language for strings of these data types shall not be supported (i.e. the class level Native Language attribute of the Parameter and Parameter Group objects).

This non-volatile instance attribute has an access rule of Get/Set.

Supported language list

List of languages supported by character strings of data type STRINGI within the device.

This volatile instance attribute has an access rule of Get.

International product name

Names of the product in various languages.

This non-volatile instance attribute has an access rule of Get.

Semaphore

Provides a semaphore for client access synchronization to the entire device.

This is a volatile attribute whose value after a reset or power-up is always zero. The Semaphore attribute can be read at any time. The value reported will be the current content of the Semaphore attribute, including the real-time (actual) timer value.

Client electronic key

It is composed of the client's Vendor ID and the client's device serial number.

Semaphore timer

This is a millisecond timer that when non-zero, counts down to zero. When the Semaphore Timer reaches the value zero, the Semaphore attribute is set to all zeros.

The Semaphore Timer component operates at the base time resolution of the device. Therefore, if the time base of the device is greater than a 1 ms resolution, the time value accepted by the attribute shall be rounded up to the next timer increment appropriate for the device.

When a Set_Attribute_Single service is directed to this attribute, the attribute will be set to the service data if either of the following is true:

- The current value in the attribute is zero;
- The Electronic Key portion of the service data matches the Electronic Key portion of the attribute data.

This instance attribute has an access rule of Get/Set.

Assigned name

This text string represents the user assigned name of the device.

This non-volatile instance attribute has an access rule of Get/Set.

Assigned description

This text string represents a user assigned description of the device and may also describe its function.

This non-volatile instance attribute has an access rule of Get/Set.

Geographic location

This text string represents the physical location of the device as provided by the user.

This non-volatile instance attribute has an access rule of Get/Set.

Type15 identity info

This attribute is reserved for Type 15 devices. Otherwise it shall not be used.

Protection mode

Indication of the present mode of protection for the device. The use of this attribute is further detailed in 6.2.1.2.2.7.

This volatile instance attribute has an access rule of Get.

Uptime

Amount of time since the device was last powered up or restarted (for example Identity object Reset). The vendor determines where in the power up/restart process this Uptime value starts being incremented, but this Uptime value shall start incrementing no later than the transition into the Standby State (see Figure 3).

This volatile instance attribute has an access rule of Get.

6.2.1.2.2.4 System management (class level) and object management (instance level)services

Object-specific details of the common services are provided below for the Identity object.

Get_Attributes_All

At the instance level, this service is **optional** for CP 2/3 else it is **mandatory**.

Reset

The Object Specific Data of the request primitive will contain a dedicated parameter to specify the type of Reset requested by the user. Its detailed format and corresponding Reset options are specified in IEC 61158-6-2.

When the Identity ASE (Object) receives a Reset request, it shall:

1) determine if it can provide the type of reset requested, and return the appropriate error if it cannot;
2) respond to the request;
3) perform the type of reset requested.

The Reset service shall not be processed when the device is in the Major Unrecoverable Fault state.

The device shall stop accepting explicit message service requests received on any port as soon as possible, but no later than 3 s of the Reset success response being sent. Messages that are not accepted may either be ignored or rejected by returning status code 0x10 (Device State Conflict). The device shall resume accepting explicit message service requests after the reset has completed.

A client that sends a Reset service to the device should wait at least s after receiving a success response before attempting to re-establish communications with the device to ensure that the response is not received from the device prior to the reset occurring. The client should be aware that devices are not constrained to complete the entire reset and resume communications within 3 s, but are only required to stop responding to service requests prior to the start of the actual reset within 3 s.

Get_Attribute_Single

At the class level, this service is **mandatory** if any attributes are implemented, else it is **optional**.

Set_Attribute_Single

This service is **mandatory** if any settable attributes are implemented (e.g. Heartbeat Interval), else it is **optional**.

Find_Next_Object_Instance

At the class level, this service is **mandatory** if non consecutive instances exist, else it is **optional**.

Get_Member

This service is **mandatory** if any attributes with the International String (STRINGI) data type are implemented, else it is **optional**.

6.2.1.2.5 Object specific services

Flash_LEDs

This service instructs the device to either start or stop flashing its Type 2-defined LEDs (used to visually identify it).

6.2.1.2.6 Identity state machine

The behavior of the Identity object is shown in the State Transition Diagram (STD) in Figure 3. This STD associates the state of the device with the status reported by the Status Attribute and with the state of the Module Status LED (see IEC 61158-4-2 for details).

A device may not be able to communicate in the Major Unrecoverable Fault state. Therefore, it might not be able to report a Major Unrecoverable Fault. It will not process a Reset service. The only exit from a Major Unrecoverable Fault is to cycle power.

The Identity object triggers production of heartbeat messages as defined by the underlying network when:

- the interval configured in the Heartbeat Interval Attribute has passed since the last heartbeat message;
- the Heartbeat message contents change, at a maximum rate of one “data changed” heartbeat message per second.

The Heartbeat Interval value shall be saved as a Non-Volatile attribute. Heartbeat messages are only triggered after the device has successfully completed the network access state machine and is online. Not all networks support sending the Heartbeat message.

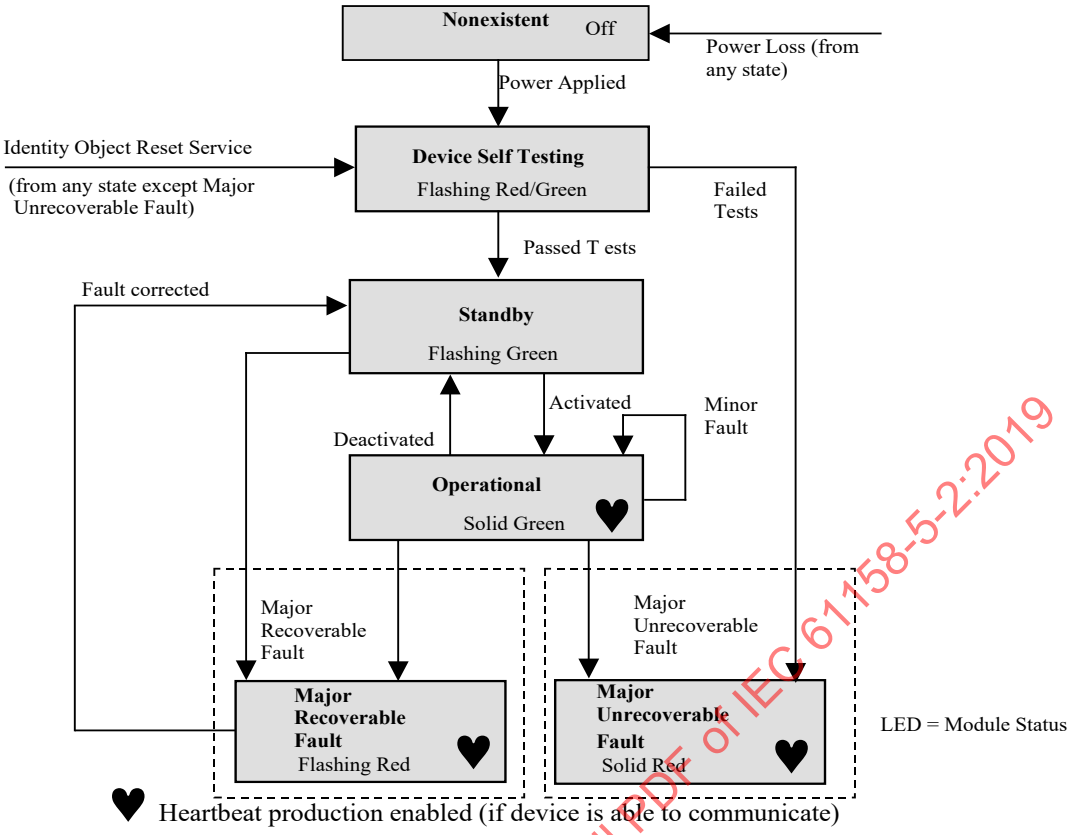


Figure 3 – Identity object state transition diagram

The STD for the Identity object contains the following events:

- Power Applied: the device is powered up;
- Passed Tests: the device has successfully passed all self tests;
- Failed Tests: the device's self test failed;
- Activated: the device's configuration is valid and the application for which the device was designed is now capable of executing (communications channels may or may not yet be established);
- Deactivated: the device's configuration is no longer valid and the application for which the device was designed is no longer capable of executing (communication channels may or may not still be established);
- Minor fault: a fault classified as either a minor unrecoverable fault or a minor recoverable fault has occurred;
- Major recoverable fault: an event classified as Major Recoverable Fault has occurred;
- Major unrecoverable fault: an event classified as a Major Unrecoverable Fault has occurred;
- Fault Corrected: the source of the Major Recoverable Fault has been corrected.

Table 7 defines the state event matrix for the Identity object.

Table 7 – Identity object state event matrix

Event	Nonexistent	Device Self Testing	Standby	Operational	Major Unrecoverable Fault	Major Recoverable Fault
Power Loss	Not Applicable	Transition to Nonexistent	Transition to Nonexistent	Transition to Nonexistent	Transition to Nonexistent	Transition to Nonexistent
Power Applied	Transition to Device Self Testing	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Failed Tests	Not Applicable	Transition to Major Unrecoverable Fault	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Passed Tests	Not Applicable	Transition to Standby	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Deactivated	Not Applicable	Ignore Event	Ignore Event	Transition to Standby	Ignore Event	Ignore Event
Activated	Not Applicable	Ignore Event	Transition to Operational	Ignore Event	Ignore Event	Ignore Event
Major Recoverable Fault	Not Applicable	Not Applicable	Transition to Major Recoverable Fault	Transition to Major Recoverable Fault	Ignore Event	Ignore Event
Major Unrecoverable Fault	Not Applicable	Not Applicable	Transition to Major Unrecoverable Fault	Transition to Major Unrecoverable Fault	Ignore Event	Ignore Event
Minor Recoverable Fault	Not Applicable	Ignore Event	Ignore Event	Ignore Event	Ignore Event	Ignore Event
Minor Unrecoverable Fault	Not Applicable	Ignore Event	Ignore Event	Ignore Event	Ignore Event	Ignore Event
Fault Corrected	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Transition to Standby
Reset	Not Applicable	Restart Self Tests	Transition to Device Self Testing	Transition to Device Self Testing	Ignore Event	Transition to Device Self Testing
Module Status LED	Off	Flashing Red/Green	Flashing Green	Solid Green	Solid Red	Flashing Red

The SEM for the Identity object contains the following states:

- Nonexistent: the device is without power;
- Device Self Testing: the device is executing its self tests;
- Standby: the device needs commissioning due to an incorrect or incomplete configuration;
- Operational: the device is operating in a fashion that is normal for the device;
- Major Recoverable Fault: the device has experienced a fault that is believed to be recoverable;
- Major Unrecoverable Fault: the device has experienced a fault that is believed to be unrecoverable.

6.2.1.2.2.7 Specific requirements for handling protection settings

Two protection settings are defined: Implicit Protection and Explicit Protection. The default value for both the Implicit and Explicit Setting is Not Protected. When both the Implicit and Explicit Protection Settings are Not Protected, the device shall accept all Type 2 explicit

messages that are valid for the device, subject to any object-specific or device-specific rules regarding state conflicts.

The Implicit Protection setting shall indicate that the device has entered a state in which it rejects explicit message requests that would disrupt its operation. The conditions under which a device enters the Implicit Protection setting are device-specific.

EXAMPLE 1

Some examples of conditions under which a device should consider entering the Implicit Protection setting are listed below:

- Presence of an established target I/O connection. This method is highly recommended for devices whose primary means of control and/or data production is via an I/O connection. For O->T connections that include the Run/Idle header, it may be further qualified that the connection be in run mode.
- The device is being controlled or run locally via a local user interface.
- The device is being controlled via an explicit message connection.

Devices shall exit the Implicit Protection setting when the conditions under which it entered the Implicit Protection setting are no longer present.

When in the Implicit Protection setting, the device shall reject those Type 2 explicit messages that would be disruptive to the device's current operation.

The following lists the disruptive explicit message requests that a device shall reject when in Implicit Protection setting:

- Reset service to the Identity Object.
- Device firmware update.
- Changes to communication settings that would cause loss of communications with the device.

Additional explicit message requests that a device may reject when in the Implicit Protection setting are device-specific. The items protected when in Implicit Protection setting are referred to as the "Implicit Protection Policy".

The Explicit Protection setting shall indicate that the device has been explicitly placed in a protected mode, either by hardware means or via a software-based setting.

EXAMPLE 2

Example methods for placing a device in Explicit Protection setting are listed below:

- Key switch on the device is placed in the "Run" position.
- Rotary switch on the module is set to a specific value to enable Explicit Protection setting.
- The device is placed in Explicit Protection setting via a local user interface (e.g., touch screen, keypad, etc.).
- The device is placed in Explicit Protection setting via secure web server interface or other secure connection means.

When in the Explicit Protection setting the device shall reject Type 2 explicit message requests that would alter the device's configuration settings or otherwise be disruptive to the device's current operation. The device may provide a (vendor-specific) mechanism for configuration of which disruptive explicit messages are to be rejected when in Explicit Protection mode. This mechanism should only be available/accessible while the device is Not Protected.

A device that implements the Protection Mode attribute may implement the Implicit Protection setting, Explicit Protection setting, or both. When both settings are implemented, the Explicit Protection setting should by default include the explicit messages covered by the Implicit Protection setting. The Implicit and Explicit Protection settings may be active simultaneously,

for example when the device has been placed in the Explicit Protection setting, and also has an I/O connection which causes disruptive services to be rejected.

Figure 4 illustrates the recommended behavior of a device that implements both the Implicit and Explicit Protection settings.

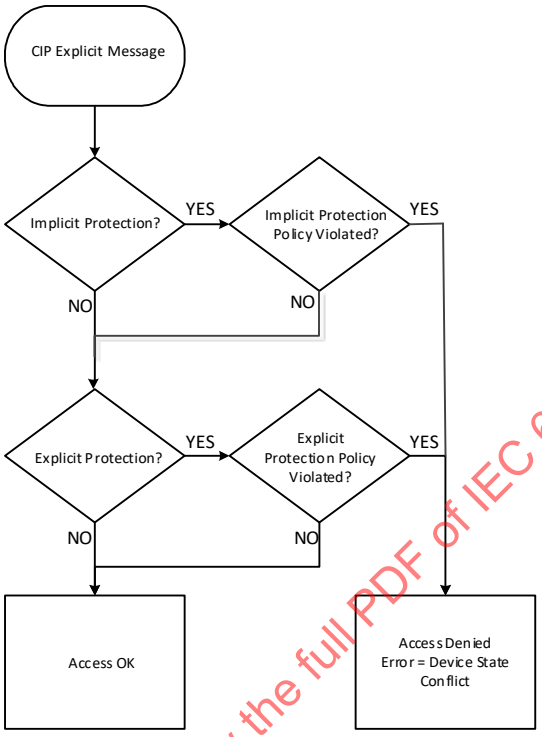


Figure 4 – Explicit and Implicit Setting interaction

Devices shall publish in user documentation the conditions under which they enter/exit either of the protection settings and what items are covered by each level supported. The items protected when in Explicit Protection setting are referred to as the “Explicit Protection Policy”.

Any explicit messages that are rejected when the device is in either of the protection levels shall be returned with a General Status code “Device State Conflict”.

6.2.1.2.3 Assembly formal model

6.2.1.2.3.1 Class definition

Each piece of data (whether real time or configuration data) communicated by a device may be represented by an attribute of one of the device internal objects. Communicating multiple pieces of data (attributes) across a single connection may require that the attributes be grouped or assembled together into a single block for optimization purpose. Instances of the Assembly object class perform this grouping. Individual components are referenced in the definitions below as "members" of the Assembly.

An Assembly object represents a buffer that binds attributes of multiple objects, allowing data to or from each object to be sent or received over a single connection. Assembly objects are used to produce and/or consume data to/from the network. An instance of the Assembly object can both produce and consume data from the network if designed to do so.

Assembly ASEs (objects) are either dynamic or static.

- Dynamic assemblies use assemblies member lists created and managed by the user. The member list may be altered by adding or deleting members;
- Static assemblies use member lists defined by the device profile or by the product implementers. The Instance number, number of members, and member list shall be fixed.

NOTE Static assemblies can usually be implemented entirely in ROM.

Instances of the Assembly object are divided into ranges specified in IEC 61158-6-2, 4.1.8.4.1.3. Dynamic assemblies shall be assigned Instance numbers in the vendor specific range.

The behavior of the Assembly object differs by the type of Assembly. A Dynamic Assembly member list is created and managed by the user of the device. The manager of the Dynamic Assembly shall specify and maintain the member list. A Static Assembly member list is defined by the device manufacturer. It shall not be modified.

To provide for the ability to create and delete objects, as well as change member lists, Dynamic Assemblies support additional services which are not supported by Static Assemblies.

(*) in front of an attribute or a service means that this attribute/service is either mandatory or optional, based on some constraints defined in the attribute/service description.

FAL ASE:			**FAL Management ASE**
CLASS:			**Assembly_Object**
CLASS ID:			**4**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
3	(m)	Attribute:	Assembly type (STATIC, DYNAMIC)
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(m)	Attribute:	Revision = 2
2	(o)	Attribute:	Max Instance
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(*)	Attribute:	Number of Members in List
2	(*)	Attribute:	Member List
2.1	(m)	Attribute:	Member Data Size
2.2	(m)	Attribute:	Member Path Size
2.3	(m)	Attribute:	Member Path
3	(m)	Attribute:	Data
4	(o)	Attribute:	Size
SYSTEM MANAGEMENT SERVICES:			
8	(c)	Constraint:	Assembly type = DYNAMIC
8.1	(m)	Mgt Service:	Create
9	(c)	Constraint:	Assembly type = DYNAMIC
9.1	(o)	Mgt Service:	Delete
14	(*)	Mgt Service:	Get_Attribute_Single
OBJECT MANAGEMENT SERVICES:			
9	(c)	Constraint:	Assembly type = DYNAMIC
9.1	(m)	Ops Service:	Delete
14	(m)	Ops Service:	Get_Attribute_Single
16	(*)	Ops Service:	Set_Attribute_Single
24	(o)	Ops Service:	Get_Member
25	(o)	Ops Service:	Set_Member

26	(c)	Constraint:	Assembly type = DYNAMIC
26.1	(*)	Ops Service:	Insert_Member
27	(c)	Constraint:	Assembly type = DYNAMIC
27.1	(*)	Ops Service:	Remove_Member

6.2.1.2.3.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.1.2.3.3 Object management attributes

Access rules (Get/Set) for the instance attributes are further limited by the operating state of the Assembly object. These additional constraints are detailed in 6.2.1.2.3.5.

Number of members in list

This instance attribute is **optional** for a Static Assembly, **mandatory** for a Dynamic Assembly.

Number of members in "Member List" attribute below.

This instance attribute has an access rule of Get only.

Member list

This instance attribute is **optional** for a Static Assembly, **mandatory** for a Dynamic Assembly.

The member list is an array of each member size and origin (internal path). Each member in the Member List contains the entries defined below.

This instance attribute has an access rule of Get only for a Static Assembly, of Set for a Dynamic Assembly.

Member data size

Size of member data (in bits).

Member path size

Size of Member Path (in octets). If no path is specified, then a value of zero shall be used.

Member path

Internal path to/from data for this member (packed format). See IEC 61158-6-2 for the format of packed path.

The following rules apply to the "Member List" attribute.

When an empty path (Member Path Size = 0) is used in an Assembly member list, the Assembly shall insert/discard the number of bits as specified in the Member Data Size sub-attribute field when producing/consuming. The Assembly object shall insert if it is producing and discards if it is consuming. The Assembly shall use a value of zero for all produced data which has been inserted.

NOTE 1 The use of an empty consumed path allows, for example, data destined for multiple nodes to be sent in a single message since each node can be configured to discard data in the message not intended for it.

The empty path shall be supported for all dynamic assemblies. Static assemblies may also contain empty paths.

No checking is required from the Assembly object at any time to verify that the size of the member data is correct for the given member path. It is the responsibility of the Assembly

member to properly handle too much or too little data. The Assembly shall deliver the configured number of bits to the member.

No padding shall be done by the Assembly object itself to align data from each Assembly member on a octet, word, or other boundary. Empty paths may be used to provide padding if desired.

Data

All of the member data packed into one array.

NOTE 2 This data can contain many different data types. For efficiency it is best to keep this data word aligned by packing it on word boundaries and adding padding as needed. This can be accomplished by using “empty paths” (Member Path Size = 0).

This instance attribute has an access rule of Set.

Size

Number of octets in "Data" attribute.

This instance attribute has an access rule of Get only.

6.2.1.2.3.4 System management (class level) and object management (instance level) services

Delete

At the instance level (object management), the Delete service deletes the specified Assembly object instance and releases all associated resources. At the class level (system management), this service deletes all existing Assembly instances.

Get_Attribute_Single

At the class level, this service is **mandatory** if any class attributes or the Member List instance attribute are implemented, else it is **optional**.

Set_Attribute_Single

The Set_Attribute_Single service modifies the contents of the specified attribute of the specified Assembly object instance. This service is **optional** for a Static Assembly, **conditional** for a Dynamic Assembly. This service may be used to add or remove a member to/from the Assembly Member List of a Dynamic Assembly. If this service is not supported for a Dynamic Assembly, then the Insert_Member and Remove_Member services shall be supported.

Get_Member

The Get_Member service returns a member from the Member List or Data instance attributes.

Set_Member

The Get_Member service modifies a member of the Member List or Data instance attributes.

Insert_Member, Remove_Member

The Insert_Member service adds a member to the Member List of a Dynamic Assembly. The Remove_Member service removes a member from the Member List of a Dynamic Assembly.

If these services are not supported for a Dynamic Assembly, then the Set_Attribute_Single service shall be supported.

6.2.1.2.3.5 Assembly state machines

Actual usage of the supported services (e.g. Get_Attribute_Single and Set_Attribute_Single) is limited by the operating state of the Assembly object. These constraints are detailed in the Assembly State Machines below.

Static Assembly state machine

Figure 5, Table 8 and Table 9 illustrate the behavior of Static Assembly objects.

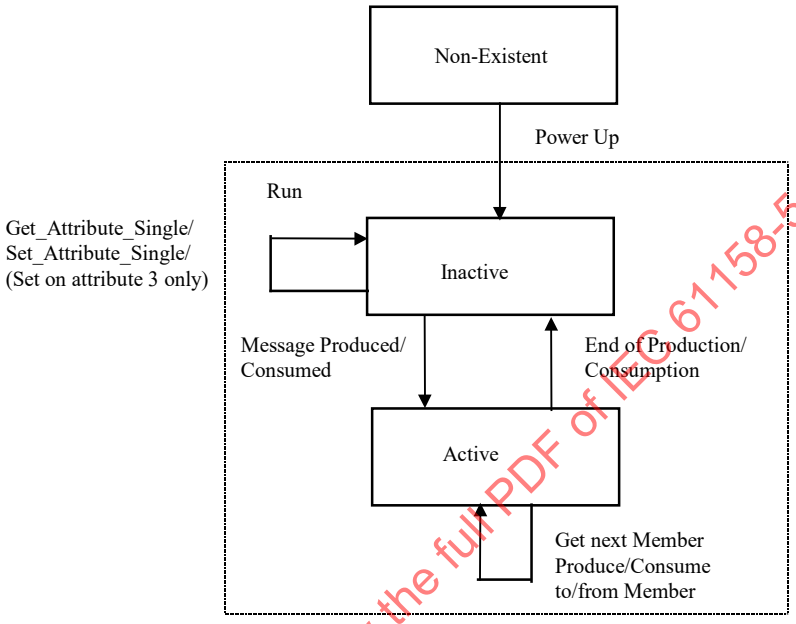


Figure 5 – Static Assembly state transition diagram

Table 8 – Static Assembly state event matrix

Event	Static Assembly object state		
	Non-existent	Inactive	Active
Power Up	Transition to Inactive	Not applicable	Not applicable
Get_Attribute_Single	Error: Object does not exist.	Validate/service the request. Return response	Validate/service the request. Return response
Set_Attribute_Single	Error: Object does not exist.	Validate/service the request. Return response	Error: Object state conflict
Message produced/ consumed	Error: Object does not exist.	Begin producing/consuming from/to each member in list. Transition to Active	Error: Object state conflict
End of production/ consumption	Error: Object does not exist.	Error: Object state conflict	Transition to Inactive

Table 9 – Static Assembly instance attribute access

Attribute	Static Assembly object state		
	Non-existent	Inactive	Active
Number of Members in List	Not available	Read only	Read only
Member List	Not available	Read only	Read only
Data	Not available	Read/Write	Read only

Dynamic Assembly state machine

Figure 6 and Table 10 illustrate the behavior of Dynamic Assembly objects.

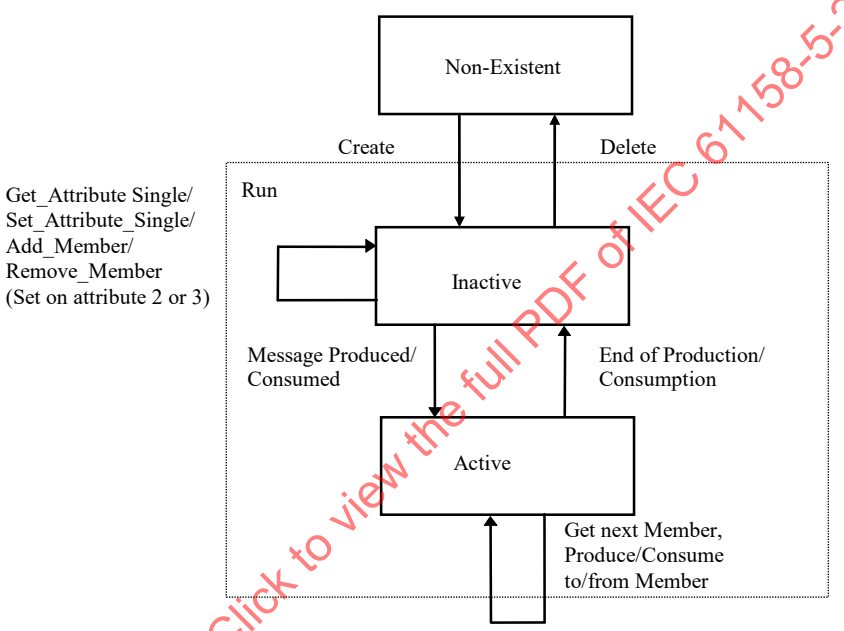


Figure 6 – Dynamic Assembly state transition diagram

Table 10 – Dynamic Assembly state event matrix

Event	Dynamic Assembly object state		
	Non-existent	Inactive	Active
Create	Class instantiates an Assembly object. Transition to Inactive	Not applicable	Not applicable
Delete	Error: Object does not exist.	Release all associated resources. Transition to Non–Existent	Error: Object state conflict
Get_Attribute_Single	Error: Object does not exist.	Validate/service the request. Return response	Validate/service the request. Return response
Set_Attribute_Single	Error: Object does not exist.	Validate/service the request. Return response	Error: Object state conflict
Insert_Member	Error: Object does not exist.	Validate/service the request. Return response	Error: Object state conflict
Remove_Member	Error: Object does not exist.	Validate/service the request. Return response	Error: Object state conflict
Message produced/ consumed	Error: Object does not exist.	Begin producing/consuming from/to each member in list. Transition to Active	Error: Object state conflict
End of production/ consumption	Error: Object does not exist.	Error: Object state conflict	Transition to Inactive

For all attributes of a Dynamic Assembly object, the Get_Attribute_Single and Set_Attribute_ Single services (when supported), are only available in the Inactive state.

Table 11 – Dynamic Assembly instance attribute access

Attribute	Dynamic Assembly object state		
	Non-existent	Inactive	Active
Number of Members in List	Not available	Read only	Read only
Member List[a]	Not available	Read/Write	Read only
Data	Not available	Read/Write	Read only

[a] This attribute can be set by either the Insert_Member service (one member at a time) or the Set_Attribute_Single service (all members at once).

6.2.1.2.3.6 Specific requirements for a connection to the Assembly object

The Assembly object shall be unique among all objects in that its connection points and instances are the same. For example, connection point 4 of the Assembly object shall be the same as instance 4 of the Assembly object. Accessing data with a path specifying "class=Assembly, instance=VV, attribute=3" shall return the same data as a path specifying "class= Assembly, connection point=VV, attribute=3".

When the Assembly object is the target of a connection, one instance plus two connection points shall be specified by the connection path in the Forward_Open service.

The format of the connection path shall be "class=Assembly, instance=XX, connection point=YY, connection point=ZZ" where XX is the instance, and YY/ZZ are the connection points. Any data segment in the path shall be used to set the data attribute of instance XX. Connection point (instance) YY shall be the consumer (O⇒T) for the connection. Likewise, connection point (instance) ZZ shall be the producer (T⇒O).

Using this format, three instances shall be specified for a connection to the Assembly object:

- configuration,
- producer,
- consumer.

For example, the following two connection paths specify the same producer instance 8. These two connections can be made simultaneously provided that the connection requested specifies multipoint.

- class=Assembly, instance= 5, connection point=4, connection point=8
- class=Assembly, instance= 3, connection point=2, connection point=8.

6.2.1.2.4 Message Router formal model

6.2.1.2.4.1 Class definition

The Message Router ASE (object) provides a formal messaging connection point through which a client may address a service to any object class or instance residing in the physical device. Every node shall contain instance 1 of the Message Router object.

The Message Router ASE (object) shall receive all incoming messages (service indications addressed to application or user objects within the device). The source of these messages may be either the Unconnected Message Manager (UCMM), or an active connection to the Message Router object.

The Message Router shall parse the first part of the message (path information) to determine the target object class.

It shall first process the Electronic Key segment and the Network segments (if present in the path). An error in the Electronic Key segment shall cause an error response to be returned with the " Key Failure in path" status.

Interpretation of the application path (e.g. class instance) shall then be performed on every service received by the Message Router. If the application path can be interpreted, the service shall then be routed internally to the target object. Any application path that cannot be interpreted by the implementation of a Message Router in a device shall cause an error response to be returned with the "Object Not Found" status.

The service indication shall then be routed to the target object for actual processing. If the service is directed to the Message Router itself (system and object management services addressing the Message Router), then the Message Router shall process the service request and respond accordingly.

When the service response has been received from the target object (or generated by the Message Router itself), then it shall be routed back to the source of the service request, using the same path in reverse . All connected service responses shall be routed back across the connection from which the service request was received. All unconnected service responses (UCMM) shall be returned via the same UCMM transaction that provided the request.

NOTE Practically, the Message Router acts as an internal switching board, which retains all the necessary routing information for messages. Therefore, the target objects do not need to be aware of this information (matching of service indications and responses is achieved via a Local ID provided by the Message Router).

FAL ASE:			**FAL Management ASE**
CLASS:			**Message_Router_Object**
CLASS ID:			**2**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(o)	Attribute:	Object_List
1.1	(m)	Attribute:	Number
1.2	(m)	Attribute:	Classes
2	(o)	Attribute:	Number available
SYSTEM MANAGEMENT SERVICES:			
1	(o)	Mgt Service:	Get_Attributes_All
14	(*)	Mgt Service:	Get_Attribute_Single
OBJECT MANAGEMENT SERVICES:			
1	(o)	Ops Service:	Get_Attributes_All
10	(o)	Ops Service:	Multiple_Service_Packet
14	(*)	Ops Service:	Get_Attribute_Single
OBJECT SPECIFIC SERVICES:			
75	(o)	Mgt Service:	Symbolic_Translation

6.2.1.2.4.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.1.2.4.3 Object management attributes

Object_List

List of object class codes supported by the device via the Message Router.

This instance attribute has an access rule of Get only.

Number

Number of supported classes in the Classes attribute below.

Classes

List of class codes supported by the device

Number available

Maximum number of connections supported by the Message Router

This instance attribute has an access rule of Get only.

6.2.1.2.4.4 System management (class level) and object management (instance level)services

Get_Attribute_Single

This service is **mandatory** if any attributes are implemented, else it is **optional**.

6.2.1.2.4.5 Object specific services

Symbolic_Translation

This service provides a translation from a Symbolic Segment path encoding to the equivalent Logical Segment path encoding, if it exists.

This service provides a mechanism for a device, that has implemented Symbolic Segment representations of internal path addresses, to translate those Symbolic Addresses encodings into their equivalent Logical Segment path encodings.

6.2.1.2.4.6 Specific requirements of the Message Router object

The Message Router object shall have one state, the Run state. It shall always be running (after power-up) and shall be fixed by device design. More information concerning the Message Router object can be found in IEC 61158-6-2 (definition of Path to access object element within a device).

The Message Router object shall support 1 or more connections to them. A Forward_Open with the parameters listed in Table 12 shall also be supported.

Table 12 – Message Router object Forward_Open parameters

Parameter	Parameter Value
O⇒T priority	low
O⇒T connection type	point-to-point
O⇒T fixed/variable	variable
O⇒T size	up to 504 octets
T⇒O priority	low
T⇒O connection type	point-to-point
T⇒O fixed/variable	variable
T⇒O size	up to 504 octets
Client/server	server
Trigger mode	ignore
Transport class	class 3 (verified)

6.2.1.2.5 Acknowledge Handler formal model

6.2.1.2.5.1 Class definition

The Acknowledge Handler object is used to manage the reception of message acknowledgments. This object communicates with a message producing application object within a device. The Acknowledge Handler object notifies the producing application of acknowledge reception; acknowledge timeouts, and production retry limit.

FAL ASE:			**FAL Management ASE**
CLASS:			**AckHandler_Object**
CLASS ID:			**43**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(m)	Attribute:	Acknowledge Timer
2	(m)	Attribute:	Retry Limit
3	(m)	Attribute:	COS Producing Connection Instance
4	(o)	Attribute:	Ack List Size
5	(o)	Attribute:	Ack List
6	(o)	Attribute:	Data with Ack Path List Size
7	(o)	Attribute:	Data with Ack Path List
SYSTEM MANAGEMENT SERVICES:			
8	(o)	Mgt Service:	Create
9	(o)	Mgt Service:	Delete
14	(o)	Mgt Service:	Get_Attribute_Single
OBJECT MANAGEMENT SERVICES:			
9	(o)	Ops Service:	Delete
14	(m)	Ops Service:	Get_Attribute_Single
16	(m)	Ops Service:	Set_Attribute_Single
OBJECT SPECIFIC SERVICES:			
75	(o)	Ops Service:	Add_AckData_Path
76	(o)	Ops Service:	Remove_AckData_Path

6.2.1.2.5.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.1.2.5.3 Object management attributes

Acknowledge timer

Time to wait for acknowledge before resending.

If the specified value for the Acknowledge Timer attribute is not equal to an increment of the available clock resolution, then the value is rounded up to the next serviceable value.

EXAMPLE A Set_Attribute_Single request is received specifying the value 5 for the Acknowledge Timer attribute and the product provides a 10 ms resolution on timers. In this case the product would load the value 10 into the Acknowledge Timer attribute.

The value that is actually loaded into the Acknowledge Timer attribute is reported in the Service Data Field of a Set_Attribute_Single response message associated with a request to modify this attribute.

This instance attribute has an access rule of Get/Set.

Retry limit

Number of Ack Timeouts to wait before informing the producing application of a RetryLimit_Reached event. A successful Set_Attribute_Single to the Retry Limit attribute will reset the Retry Counter.

This instance attribute has an access rule of Get/Set (Set optional).

COS producing connection instance

Connection instance which contains the path of the producing I/O application object which will be notified of Ack Handler events.

This instance attribute has an access rule of Get only when the Acknowledge Handler object instance is active (at least one member in the Ack List). It has an access rule of Get/Set when the Acknowledge Handler object instance is inactive.

Ack list size

Maximum number of members in Ack List.

This instance attribute has an access rule of Get only.

Ack list

List of active connection instances which are receiving Acks. The Ack List attribute is updated when an associated connection transitions between configuring, established, timed-out, and non-existent. See the state event matrix in 6.2.1.2.5.6 for details.

This instance attribute has an access rule of Get only.

Data with ack path list size

Maximum number of members in Data with Ack Path List.

This instance attribute has an access rule of Get only.

Data with ack path list

List of connection instance/consuming application object pairs. This attribute is used to forward data received with acknowledgment.

This instance attribute has an access rule of Get only.

6.2.1.2.5.4 System management (class level) and object management (instance level) services

Create

At the instance level (object management), the Create service creates an Acknowledge Handler object.

Delete

At the instance level (object management), the Delete service deletes the specified Acknowledge Handler object. At the class level (system management), this service deletes all dynamically created Acknowledge Handler instances.

6.2.1.2.5.5 Object specific services

Add_AckData_Path

This service adds a path for data with acknowledgment for a connected consumer.

Remove_AckData_Path

This service removes a path for data with acknowledgement for the given connected consumer.

6.2.1.2.5.6 Acknowledge Handler state machines

Table 13 is the state event matrix for the Acknowledge Handler object associated with a Change of State connection. Table 14 is the state event matrix for the producing I/O application object.

Table 13 – Acknowledge Handler object state event matrix

Event	Acknowledge Handler object state		
	Non-existent	**Inactive**	**Active**
Receive Acknowledge	Not applicable	Ignore event	1) Clear Ack Flag 2) Forward any data to application object IF all acknowledges received: 3a) Clear Ack Timer and Retry Counter 3b) Send Acknowledge_Received event 3c) message to producing application object
Acknowledge Timer expires	Not applicable	Ignore event	Send Ack_Timeout event message to producing application object.
Data sent	Not applicable	Ignore event	1) Set Ack Required Flag 2) Set Ack Timer 3) Clear Retry Counter
Data resent	Not applicable	Ignore event	1) Set Ack Required Flag & Ack Timer IF Retry_Limit > 0 2a) Increment Retry Counter 2b) IF Retry Counter = Retry_Limit Send Rety_Limit_Reached event message to producing application object
Delete	Not applicable	Transition to Non-Existent	Transition to Non-Existent
Create	Transition to inactive	Not applicable	Not applicable
Apply attributes	Not applicable	Verify new connection can be added to list. Pass this message to the consumed application object, if one is configured for this connection.	Verify new connection can be added to list. Pass this message to the consumed application object, if one is configured for this connection.
Connection transition to Established	Not applicable	Add this connection instance to the connection list (or internally flag as ‘Acking’) . Pass this message to the consumed application object, if one is configured for this connection. Send Acknowledge_Active event message to producing application. Transition to Active.	Add this connection instance to the connection list. Pass this message to the consumed application object, if one is configured for this connection.
Inactivity/Watchdog Timer expires	Not applicable	Not applicable	1) Internally flag this connection as ‘Not Acking’. An acknowledgement will no longer be monitored for this connection, however it remains in the connection list. 2) Pass this event to the consumed application object, if one is configured for this connection. 3) If no ‘Acking’ connections in list, send Acknowledge_Inactive event to producing application and transition to Inactive.

Event	Acknowledge Handler object state		
	Non-existent	Inactive	Active
Connection deleted	Not applicable	Ignore event	1) Remove this connection instance from the connection list. 2) Pass this event to the consumed application object, if one is configure for this connection. 3) If no 'Acking' connections in list, send Acknowledge_Inactive event to producing application and transition to Inactive.

Table 14 – Producing I/O application object state event matrix

Event	Producing I/O application object state			
	Not running	Running	Running with acknowledgment	Prohibited
Change of state detected	Ignore event	1) Inform Link Producer to Send Data. IF Inhibit Time configured: 2a) Start Inhibit Timer 2b) Transition to Produced	1) Inform Link Producer to Send Data. 2) Send DataSent event message to Acknowledge Handler object. IF Inhibit Time configured 3a) Start Inhibit Timer 3b) Transition to Prohibited.	Queue event
Acknowledge received	Not applicable	Not applicable	Ignore event	IF Ack_Active Set 1a) IF Inhibit Timer not running Transition to Running with Acknowledgement 1b) ELSE Set Ack_Receive flag ELSE ignore event
Acknowledge timeout	Ignore event	Not applicable	1) Inform Link Producer to send data 2) Send Data_Resent event message to Acknowledge Handler object.	1) Inform Link Producer to send data. 2) Send Data_Resent event message to Acknowledge Handler object.
Transmission timer expires	Not applicable	Inform Link Producer to send data.	1) Inform Link Producer to send data. 2) Send Data_Sent event message to Acknowledge Handler object.	1) Inform Link Producer to send LAST data sent. 2) Send Data_Sent event message to Acknowledge Handler object.
Retry limit reached	Ignore event	Not applicable	Product specific	Transition to Running with Acknowledgement.
Inhibit timer expires	Ignore event	Not applicable	Not applicable	IF Ack_Active set 1a) IF Ack_Received Transition to Running w/Ack Clear Ack_Received flag 1b) ELSE Ignore event ELSE Transition to Running
Connection deleted	Not applicable	Transition to Not Running	Transition to Not Running	Transition to Not Running

Event	Producing I/O application object state			
	Not running	Running	Running with acknowledgment	Prohibited
Acknowledge active	Set Ack Active Flag	1) Transition to Running with Acknowledgement 2) Set Ack Active Flag	Ignore event	Set Ack_Active Flag
Acknowledge Inactive	Reset Ack Active Flag	Ignore event	1) Transition to Running 2) Reset Ack Active Flag	Reset Ack_Active Flag
Connection transition to Established	IF Ack Active Transition to Running with Acknowledg-ment IF Ack Inactive Transition to Running	Ignore event	Ignore event	Ignore event
NOTE This is a partial state event matrix for a Producing I/O Application object. Only those states and events associated with data acknowledgement are defined. Other states and events will most likely be associated with a producing I/O application object.				

6.2.1.2.5.7 Specific requirements for behavior and configuration

Behavior and configuration of acknowledged data production

The following rules are used to configure and determine the behavior of an acknowledged Change of State or Cyclic I/O connection using the Acknowledge Handler object. In the following examples, COS Producer is used to reference the device producing change of state or cyclic data and consuming an acknowledgment (client). A COS Consumer is used to reference the device consuming the change of state or cyclic data and producing an acknowledgment (server).

Acknowledged data production

a) The COS Producers consumed connection path shall be set to an available Acknowledge Handler object. The path shall consist of Class and Instance. If an Acknowledge Handler object is not available, use the Acknowledge Handler Class Create service to obtain a new one.

b) The COS Producers producing I/O application informs the Acknowledge Handler object of new data production (Data Sent event message) or data production retries (Data Resent event message).

c) The COS Producers acknowledgment reception is done by the Acknowledge Handler object. The Acknowledge Handler object informs the Producing I/O application when one or more acknowledges have not been received within the acknowledge timeout (using the Ack List and Ack Timeout attributes).

d) The acknowledge message requires no data. The COS producing device's Acknowledge Handler object shall consider valid message reception as an acknowledgment. However, a change of state or cyclic producing device may be configured to consume data along with the acknowledgment. In this case the data is forwarded to the application object in the "Data with Ack Path List" attribute, based on the connection which received the data.

e) The COS Consumer acknowledge producing application shall be configured to send either a zero length message or valid response (output) message when a valid input message is consumed.

f) An acknowledge timer is started each time production occurs. The Acknowledge Handler object is notified of this event by a Data Sent or Data Resent event message from the producing application.

g) Expiration of the acknowledge timer causes an Acknowledge Timeout message to be sent to the producing application object. That object shall resend the last message if the Retry Limit has not been reached. It may also take an application specific action.

h) The retry count is incremented each time an Acknowledge Timeout message is sent to the producing application. When the retry limit has been reached, a Retry Limit Reached message is sent to the producing application object.

i) The retry count is cleared on each Data Sent message. A Data Resent message does not clear the retry counter.

j) The acknowledge timer value is configurable within the Acknowledge Handler object.

k) The number of retries is (optionally) configurable within the Acknowledge Handler object.

Use of timers with acknowledged data production

The following rules shall be observed when sending acknowledged data.

l) New data not sent while the Inhibit Timer is active (running).

m) New data is sent when no acknowledge is pending, subject to rule # a (an acknowledge is pending after a send of new data or a retry of old data and until an Ack Timeout or Ack Received).

n) Retry of old data occurs at Ack Timeout if new data is not available or the Inhibit Timer is active.

o) Sending new data (or old data on transmission trigger timeout) starts the Ack Timer, Inhibit Timer, and the Transmission Trigger Timer. The Retry Counter is also cleared.

p) A retry of old data starts the Ack Timer.

Figure 7 shows the typical relationship of timers within the Acknowledge Handler object. An example of the typical timing relationships is shown in Figure 8.

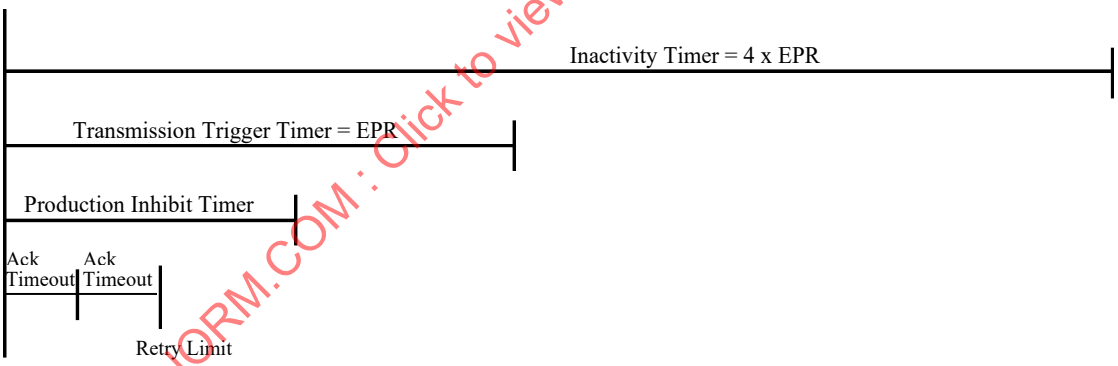


Figure 7 – Typical timing relationships for acknowledged data production

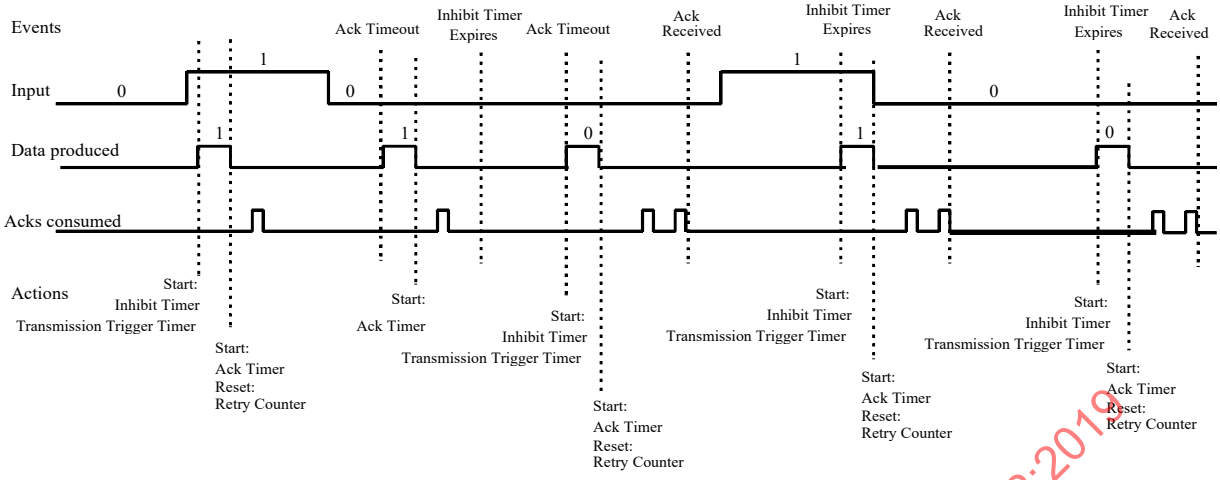


Figure 8 – Example of a COS system with two acking devices

The following diagrams illustrate the message flow in a Change of State connection for both single (see Figure 9) and multi-consumer configurations (see Figure 10).

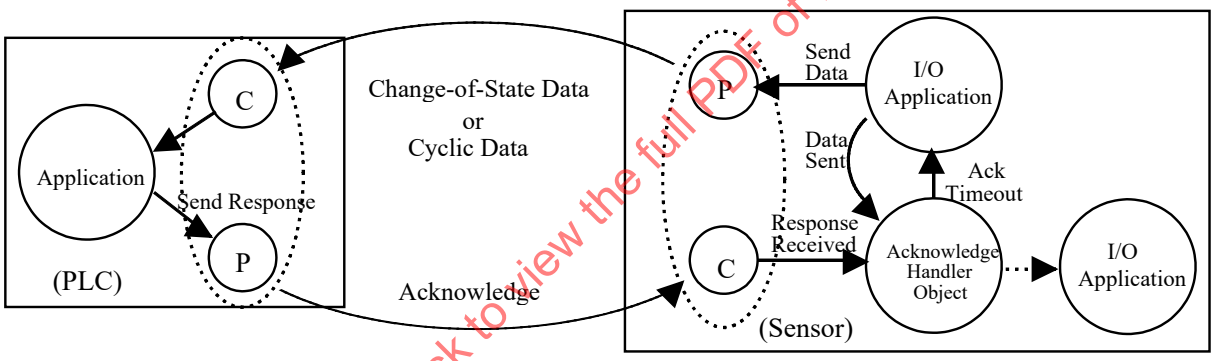


Figure 9 – Message flow in COS connection – one Connection object, one consumer

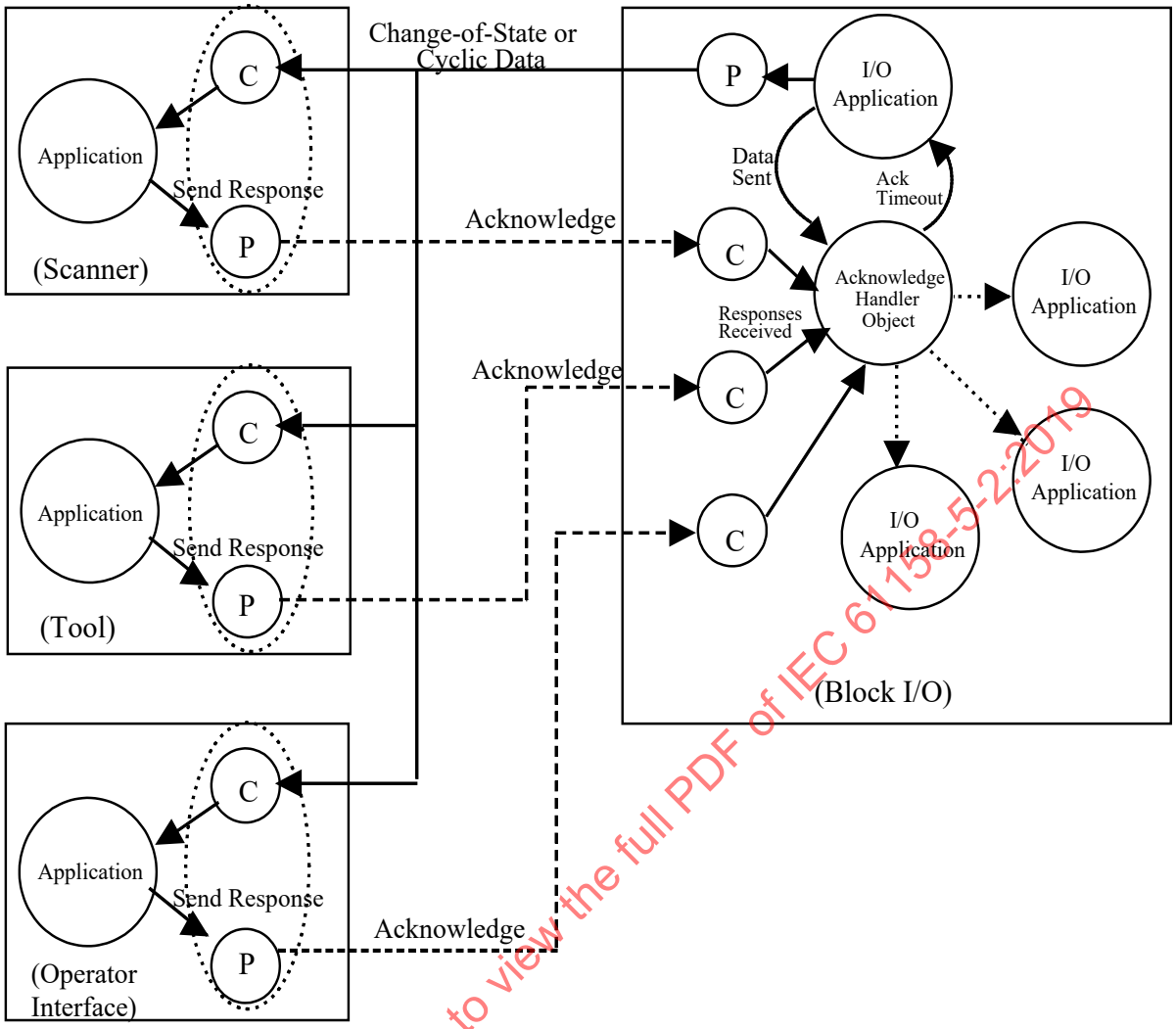


Figure 10 – Message flow in COS connection – multiple consumers

6.2.1.2.6 Time Sync formal model

6.2.1.2.6.1 Class definition

The Time Sync ASE (object) provides an IEC 61158 Type 2 interface to the IEC 61588:2009 Precision clock synchronization protocol for networked measurement and control systems, commonly referred to as the Precision Time Protocol (PTP). Any device supporting Type 2 Time Synchronization shall provide a single instance (instance 1) of the Time Sync object.

NOTE Additional details can be found in IEC 61588:2009.

This object provides attributes and services to:

- Get clock status and properties such as synchronized state, current offset to master, and grandmaster identity;
- Access PTP clock management functions such as clock priority;
- Access the PTP network of devices via native PTP management messages.

(*) in front of an attribute or a service means that this attribute/service is either mandatory or optional, based on some constraints defined in the attribute/service description.

FAL ASE: **FAL Management ASE**
CLASS: **Time_Sync_Object**
CLASS ID: **67**
PARENT CLASS: **Base_Object**
ACCESS ATTRIBUTES:

1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):

1	(m)	Attribute:	Revision = 3
2	(o)	Attribute:	Max Instance
3	(o)	Attribute:	Number of Instances
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):

1	(m)	Attribute:	PTPEnable
2	(m)	Attribute:	IsSynchronized
3	(m)	Attribute:	SystemTimeMicroseconds
4	(m)	Attribute:	SystemTimeNanoseconds
5	(m)	Attribute:	OffsetFromMaster
6	(m)	Attribute:	MaxOffsetFromMaster
7	(m)	Attribute:	MeanPathDelayToMaster
8	(m)	Attribute:	GrandMasterClockInfo
8.1	(m)	Attribute:	ClockIdentity
8.2	(m)	Attribute:	ClockClass
8.3	(m)	Attribute:	TimeAccuracy
8.4	(m)	Attribute:	OffsetScaledLogVariance
8.5	(m)	Attribute:	CurrentUtcOffset
8.6	(m)	Attribute:	TimePropertyFlags
8.7	(m)	Attribute:	TimeSource
8.8	(m)	Attribute:	Priority1
8.9	(m)	Attribute:	Priority2
9	(m)	Attribute:	ParentClockInfo
9.1	(m)	Attribute:	ClockIdentity
9.2	(m)	Attribute:	PortNumber
9.3	(m)	Attribute:	ObservedOffsetScaledLogVariance
9.4	(m)	Attribute:	ObservedPhaseChangeRate
10	(m)	Attribute:	LocalClockInfo
10.1	(m)	Attribute:	ClockIdentity
10.2	(m)	Attribute:	ClockClass
10.3	(m)	Attribute:	TimeAccuracy
10.4	(m)	Attribute:	OffsetScaledLogVariance
10.5	(m)	Attribute:	CurrentUtcOffset
10.6	(m)	Attribute:	TimePropertyFlags
10.7	(m)	Attribute:	TimeSource
11	(m)	Attribute:	NumberOfPorts
12	(m)	Attribute:	PortStateInfo
12.1	(m)	Attribute:	NumberOfPorts

12.2.1	(m)	Attribute:	PortNumber
12.2.2	(m)	Attribute:	PortState
13	(m)	Attribute:	PortEnableCfg
13.1	(m)	Attribute:	NumberOfPorts
13.2.1	(m)	Attribute:	PortNumber
13.2.2	(m)	Attribute:	PortEnable
14	(m)	Attribute:	PortLogAnnounceInervalCfg
14.1	(m)	Attribute:	NumberOfPorts
14.2.1	(m)	Attribute:	PortNumber
14.2.2	(m)	Attribute:	PortlogAnnounceInterval
15	(m)	Attribute:	PortLogSyncIntervalCfg
15.1	(m)	Attribute:	NumberOfPorts
15.2.1	(m)	Attribute:	PortNumber
15.2.2	(m)	Attribute:	PortLogSyncInterval
16	(*)	Attribute:	Priority1
17	(*)	Attribute:	Priority2
18	(m)	Attribute:	DomainNumber
19	(m)	Attribute:	ClockType
20	(m)	Attribute:	ManufactureIdentity
21	(m)	Attribute:	ProductDescription
21.1	(m)	Attribute:	Size
21.2	(m)	Attribute:	Description
22	(m)	Attribute:	RevisionData
22.1	(m)	Attribute:	Size
22.2	(m)	Attribute:	Revision
23	(m)	Attribute:	UserDescription
23.1	(m)	Attribute:	Size
23.2	(m)	Attribute:	Description
24	(m)	Attribute:	PortProfileIdentityInfo
24.1	(m)	Attribute:	NumberOfPorts
24.2.1	(m)	Attribute:	PortNumber
24.2.2	(m)	Attribute:	PortProfileIdentity
25	(m)	Attribute:	PortPhysicalAddressInfo
25.1	(m)	Attribute:	NumberOfPorts
25.2.1	(m)	Attribute:	PortNumber
25.2.2	(m)	Attribute:	PhysicalProtocol
25.2.3	(m)	Attribute:	SizeOfAddress
24.2.4	(m)	Attribute:	PortPhysicalAddress
26	(m)	Attribute:	PortProtocolAddressInfo
26.1	(m)	Attribute:	NumberOfPorts
26.2.1	(m)	Attribute:	PortNumber
26.2.2	(m)	Attribute:	NetworkProtocol
26.2.3	(m)	Attribute:	SizeOfAddress
26.2.4	(m)	Attribute:	PortProtocolAddress
27	(m)	Attribute:	StepsRemoved
28	(m)	Attribute:	SystemTimeAndOffset
28.1	(m)	Attribute:	SystemTime
28.2	(m)	Attribute:	SystemOffset
29	(*)	Attribute:	AssociatedInterfaceObjects
29.1	(m)	Attribute:	NumberOfPorts
29.2.1	(m)	Attribute:	PortNumber
29.2.2	(m)	Attribute:	AssociatedObjectPathSize
29.2.3	(m)	Attribute:	AssociatedObject

SYSTEM MANAGEMENT SERVICES:

1	(o)	Mgt Service:	Get_Attributes_All
14	(m)	Mgt Service:	Get_Attribute_Single

OBJECT MANAGEMENT SERVICES:

3	(m)	Ops Service:	Get_Attribute_List
4	(m)	Ops Service:	Set_Attribute_List
14	(m)	Ops Service:	Get_Attribute_Single
16	(m)	Ops Service:	Set_Attribute_Single

6.2.1.2.6.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.1.2.6.3 Object management attributes

PTPEnable

Specifies whether the Precision Time Protocol is enabled for this device.

PTPEnable does not apply to the transparent clock function(s) within a device. If the transparent clock functions of a device are configurable, they are managed through vendor specific means.

The default values specified in Table 15 were chosen to simplify initial system construction for most cases. There are conflicting requirements for devices implementing a boundary clock. For instance, if the device is designed to serve the role of a CIP Sync master (e.g. a controller with multiple PTP ports), then it should default to Disabled. However, when possible, boundary clocks should default to Enabled.

Table 15 – PTPEnable attribute default values

PTP Clock type	Default value
Slave-only Ordinary Clock	Enabled
Boundary Clock	Product specific, Enabled preferred
Master-capable Ordinary Clock	Disabled
Transparent Clock	Not applicable

This non-volatile instance attribute has an access rule of Get/Set.

However, modular devices with a boundary clock may treat PTPEnable as volatile if another module provides the value.

IsSynchronized

Specifies whether the local clock is synchronized with a master reference clock. At a minimum, a clock is synchronized if it has one port in the slave state and is receiving updates from the time master. A particular device may impose additional criteria or constraints on synchronization which are outside the scope of this document.

EXAMPLE A device can specify a synchronization threshold of 10 µs and only indicate that the device is synchronized when the OffsetFromMaster (attribute 5) is less than 10 µs for a period of 5 sync intervals (attribute 15).

This volatile instance attribute has an access rule of Get only.

SystemTimeMicroseconds

Specifies the current system time in units of microseconds. See 6.2.1.2.6.5 (System Time definition).

This volatile instance attribute has an access rule of Get/Set. Set is mandatory for master clock capable devices and is optional otherwise.

SystemTimeNanoseconds

Same as SystemTimeMicroseconds except in units of nanoseconds.

This volatile instance attribute has an access rule of Get/Set. Set is mandatory for master clock capable devices and is optional otherwise.

OffsetFromMaster

Specifies the amount of deviation between the local clock and its master clock in nanoseconds.

This volatile instance attribute has an access rule of Get only.

MaxOffsetFromMaster

Specifies the maximum amount of deviation between the local clock and the master clock in nanoseconds since last set. The value may be represented as a signed or absolute value. It is settable, typically to 0.

This volatile instance attribute has an access rule of Get/Set.

MeanPathDelayToMaster

Specifies the average path delay between the local clock and master clock in nanoseconds.

This volatile instance attribute has an access rule of Get only.

GrandmasterClockInfo, ParentClockInfo, LocalClockInfo

GrandmasterClockInfo, ParentClockInfo, and LocalClockInfo specify clock property information for the Grandmaster, Parent and Local PTP clock respectively. The data is extracted from the PTP data sets maintained by the PTP subsystem.

The ClockIdentity provides a unique identifier for the clock. The clock time source, class, variance, and other attributes provide additional information about the properties of the clock.

These volatile instance attributes have an access rule of Get only.

ClockIdentity

Specifies the unique identifier for the clock. The format of the identifier depends on the network protocol. CP2/2 encodes the MAC address into the identifier. CP2/3 and CP2/1 encode the Vendor ID and Serial Number.

PortNumber

Specifies the port number of the port identity.

ClockClass

Specifies the class of the clock quality. The clock class represents a relative measure of the clock quality used by the Best Master algorithm to determine the grandmaster. The class is a value between 0 and 255, with 0 as the best clock.

TimeAccuracy

Specifies the expected absolute accuracy of the clock relative to the PTP epoch. TimeAccuracy is the accuracy measure of clock quality used by the Best Master algorithm to determine the grandmaster. The accuracy is specified as a graduated scale starting at 25 ns (nanoseconds) and ending at greater than 10 s (seconds) or unknown. A GPS time source will have an accuracy of approximately 250 ns (nanoseconds). A Hand_Set clock will typically have accuracy less than 10 s (seconds). The lower the accuracy value, the better the clock.

OffsetScaledLogVariance

Specifies a measure of the inherent stability properties of the clock. OffsetScaledLogVariance is the variance measure of clock quality used by the Best Master algorithm to determine the grandmaster. The value is represented in offset scaled log units. The lower the variance, the better the clock.

CurrentUtcOffset

Specifies the current UTC offset in seconds from International Atomic Time (TAI) of the clock. As of 2006-01-01 at 00:00:00 UTC, the offset was 33 s (seconds).

TimePropertyFlags

Specifies the time property flags of the clock.

TimeSource

Specifies the primary time source of the clock.

Priority1 and Priority2

Specify the relative priority of the grandmaster clock to other clocks in the system. See Priority1 and Priority2 main attributes 16 and 17 respectively.

ObservedOffsetScaledLogVariance

Specifies an estimated measure of the parent clock's variance as observed by the slave clock.

ObservedPhaseChangeRate

Specifies an estimated measure of the parent clock's drift as observed by the slave clock.

NumberOfPorts

Specifies the number of PTP ports on the device. PTP Ordinary clocks have one port. PTP Boundary and Transparent clocks have more than one port. A hybrid clock that contains both an ordinary clock and an end-to-end transparent clock has a value of one (1) for this attribute.

This volatile instance attribute has an access rule of Get only.

Port stateInfo

Specifies the current status of each PTP port on the device.

This volatile instance attribute has an access rule of Get only.

PortEnableCfg

Specifies the port enable configuration of each port on the device.

This non-volatile instance attribute has an access rule of Get/Set.

PortLogAnnounceIntervalCfg

Specifies the PTP announce interval between successive “Announce” messages issued by a master clock on each PTP port of the device. The units of the PortLogAnnounceInterval member are log base 2 s (seconds). See 6.2.1.2.6.5 (CPF 2 PTP profile default settings and ranges) for default values.

This structure non-volatile instance attribute has an access rule of Get/Set.

PortLogSyncIntervalCfg

Specifies the PTP sync interval between successive “Sync” messages issued by a master clock on each PTP port of the device. The units of the PortLogSyncInterval member are log base 2 s (seconds). See 6.2.1.2.6.5 (CPF 2 PTP profile default settings and ranges) for default values.

This structure non-volatile instance attribute has an access rule of Get/Set.

Priority1

Specifies the Priority1 setting of the PTP Best Master Algorithm. This attribute specifies the Best Master ranking of this clock and supersedes the clock quality (class, accuracy, and variance). This attribute allows the user to override the automatic selection of the best master clock before any quality measures are evaluated. The value is between 0 and 255. The highest priority is 0. See 6.2.1.2.6.5 (CPF 2 PTP profile default settings and ranges) for default values.

This instance attribute is mandatory for master capable devices, it is optional for all others.

This non-volatile instance attribute has an access rule of Get/Set.

Priority2

Specifies the Priority2 setting of the PTP Best Master Algorithm. This attribute specifies the Best Master ranking of this clock after clock quality (class, accuracy, and variance) has been evaluated and supersedes the tie-breaker. This attribute allows the user to override the automatic selection of the best master clock after quality measures have been evaluated, i.e. choose the best master from a set of clocks of equal quality. The value is between 0 and 255. The highest priority is 0. See 6.2.1.2.6.5 (CPF 2 PTP profile default settings and ranges) for default values.

This instance attribute is mandatory for master capable devices, it is optional for all others.

This non-volatile instance attribute has an access rule of Get/Set.

DomainNumber

Specifies the PTP clock domain. See 6.2.1.2.6.5 (CPF 2 PTP profile default settings and ranges) for default values.

This non-volatile instance attribute has an access rule of Get/Set.

ClockType

Specifies the PTP functions that the node supports. A PTP node may implement more than one type of clock (e.g. an ordinary clock may be combined with an end-to-end transparent clock). A bit index set to one indicates the clock type is supported.

This volatile instance attribute has an access rule of Get only.

ManufactureIdentity

Specifies the manufacturer identity of the clock. The first 3 octets specify the IEEE OUI (Organization Unique Id) for the manufacturer. The last octet is reserved.

This volatile instance attribute has an access rule of Get only.

ProductDescription

Specifies the product description of the device that contains the clock. The format is:

- name of manufacturer of the device followed by a semicolon;
- model number of the device followed by a semicolon;
- serial number.

This volatile instance attribute has an access rule of Get only.

RevisionData

Specifies the revision data of the device that contains the clock. The format is:

- hardware revision of the clock followed by a semicolon;
- firmware revision of the clock followed by a semicolon;
- software revision of the clock.

Subfields that do not apply may be null or blank (e.g. “1.2;2.3;” or “;;3.4”).

This volatile instance attribute has an access rule of Get only.

UserDescription

Specifies the user description of the device that contains the clock. The format is:

- user defined name or description of the device followed by a semicolon;
- user defined physical location of the device.

This volatile instance attribute has an access rule of Get only.

PortProfileIdentityInfo

Specifies the PTP profile of each port of the device. The attribute returns the profile identity of the currently active profile.

This volatile instance attribute has an access rule of Get only.

PortPhysicalAddressInfo

Specifies the physical protocol and physical address of each port of the device.

This volatile instance attribute has an access rule of Get only.

PortProtocolAddressInfo

Specifies the network protocol and protocol address of each port of the device (e.g. IP address). NetworkProtocol specifies the protocol for the network.

This volatile instance attribute has an access rule of Get only.

StepsRemoved

Specifies the number of communication paths traversed between the local clock and the grandmaster clock.

This instance attribute has an access rule of Get only.

SystemTimeAndOffset

Specifies the System Time in microseconds and the Offset to the local clock value. The responding device will return the current System Time and Offset. See specific "clock model" in 6.2.1.2.6.5.

This volatile instance attribute has an access rule of Get only.

AssociatedInterfaceObjects

Specifies the objects associated with each PTP port of the device.

PTP port numbers shall start with 1 and be sequential, according to IEC 61588. When it is not possible for the PTP and CPF 2 port numbers to be the same, or when the PTP port is associated with a physical port, the device needs to identify the associations. The AssociatedInterfaceObjects attribute specifies for each PTP port whether it is associated with a CPF 2 port or a physical port. See 6.2.1.2.6.5 c) (PTP port assignments) for examples where the PTP port and CPF 2 port numbers may not be the same.

If the PTP Port is associated with a CPF 2 port, the AssociatedObject shall specify the object instance that represents the CPF 2port. That is, the AssociatedObject shall specify the Port object instance (20 F4 24 xx, where xx is the Port object instance).

If the PTP Port is associated with a physical Ethernet port (for example in the PRP case), the AssociatedObject shall specify the Ethernet Link object instance (20 F6 24 xx, where xx is the Ethernet Link object instance).

This instance attribute is mandatory if the PTP port numbers do not match the CPF 2 port numbers, otherwise it is optional.

This non-volatile instance attribute has an access rule of Get only.

6.2.1.2.6.4 System management (class level) and object management (instance level services

No specific requirement for this object.

6.2.1.2.6.5 Specific behavior for the Time Sync object

a) System Time definition

The starting date/time for CPF2 time synchronization System Time is 1970-01-01 at 00:00:00. System Time is represented as a 64-bit value (ULINT) in mircroseconds (attribute SystemTimeMicroseconds) or nanoseconds (attribute SystemTimeNanoseconds). System Time is adjusted for leap seconds, but not local time zones or daylight savings time. It represents the current time at the 0th meridian.

In the Precision Time Protocol, time is distributed as the number of nanoseconds since the beginning of the PTP epoch, 1969-12-31 at 23:59:51.999918 UTC. Leap seconds are distributed by the Precision Time Protocol as the "current UTC offset". As of 2006-01-01 at 00:00:00 UTC, the number of leap seconds was 33.

PTP time and leap seconds are converted to microseconds (attribute SystemTimeMicroseconds) or nanoseconds (attribute SystemTimeNanoseconds) to give System Time as:

SystemTime = PTPTime – CurrentUtcOffset

b) Clock identity

Each PTP clock shall assign a unique 8-octet identifier for the clock. The assignment of identifiers depends on the CPF 2 network. For CP 2/2 the identifiers are based on the MAC address. For other CPF 2 networks, the identifiers are based on the Vendor Id and Serial Number. A local bus may also implement the PTP protocol and assign the identifier using the encoding for a local or closed network.

The identifiers are formatted as specified in IEC 61158-6-2, Table 111.

c) PTP port assignments

Each device network port is given a unique PTP port number according to IEC 61588. The PTP port numbers shall start with 1 and be sequential. In order to be consistent with the CPF 2 protocols, the PTP and CPF 2 port numbers should be the same where possible.

The following are some reasons why the PTP and CPF 2 port numbers cannot be the same:

- The device might not number its CPF 2 port numbers sequentially starting with 1. For example, the device might not support CPF 2 port number 1 (backplane port).
- For devices that support PRP, there shall be two PTP ports (one per physical Ethernet port), but one CPF 2 port.

See 6.2.1.2.6.3 for a description of the attribute to define the PTP port relationship.

d) CPF 2 PTP profile

As defined by IEC 61588, the purpose of a PTP profile is to allow organizations to specify unique selections of attribute values and optional features of the protocol that, when using the same transport protocol, will inter-work and achieve a performance that meets the requirements of a particular application.

The PTP profile for the CPF 2 networks is based on the Delay Request-Response default PTP profile defined in IEC 61588.

NOTE The identification and default settings of the CPF 2 PTP profile will be the same as the IEC 61588 PTP default specification.

e) Profile identification

Table 16 identifies the CPF 2 PTP profile.

Table 16 – Profile identification

Profile identifier	Specification
PTP profile description	CIP Sync Profile
Version	1.0
Profile identifier	00-21-6C-00-01-00
Specifying organization	ODVA: Web: www.odva.org

f) CPF 2 PTP profile default settings and ranges

The CPF 2 profile shall support the PTP default and range settings as defined in Table 17. The table defines the required range settings for this profile, but a device may support a larger range (e.g. Log sync interval -3 to +2) as long as those values fall within the range acceptable values for IEC 61588.

Table 17 – Profile default settings and ranges

PTP Attribute	Attribute ID	Default [a]	Required range	Acceptable range	Units
Log announce interval	14	1	0 to 4	0 to 4	Log_2 seconds
Log sync interval	15	0	-1 to +1	-3 to +2	Log_2 seconds
Priority1	16	128	0 to 255	0 to 255	
Priority2	17	128	0 to 255	0 to 255	
Domain	18	0	0 to 127	0 to 127	
Announce receipt timeout interval		3	3	2 to 10	Announce Intervals

[a] The profile default settings may differ for ports connected to a proprietary backplane or bus.

g) Profile transports

Table 18 identifies the IEC 61588 transport specified for each Type 2 network and a corresponding reference in the IEC 61588 specification.

Table 18 – Profile transports

Network	IEC 61588 transport	IEC 61588 reference
CP 2/1	Transport of PTP over CP 2/1	Annex H
CP 2/2	Transport of PTP over UDP/IPv4	Annex D
CP 2/3	Transport of PTP over CP 2/3	Annex G

h) PTP domain

A CPF 2 network shall be limited to one PTP domain.

The IEC 61588 specification allows for the presence of more than one clock domain on a network. However, for some implementations of PTP, based on IEC 61588:2004 (version 1) hardware, the presence of other clock domains will cause interference. Therefore, to avoid interference problems a CPF 2 network will be limited to one domain. It is recommended that new hardware implementations be designed to tolerate the presence of multiple domains.

i) PTP node management

Each device implementing CPF 2 PTP profile shall implement an instance of the Time Sync object. In addition the device shall implement the management message mechanism defined in IEC 61588.

j) Transparent Clock

Devices with multiple ingress/egress ports shall implement an end-to-end transparent clock between those ports.

k) Path delay mechanism

Each device implementing CPF 2 PTP profile shall implement the delay request-response path delay measurement mechanism.

l) Recommended PTP clock settings

Table 19 defines recommended default settings for PTP clock attributes. These settings provide general recommendations for various types of devices to assign a relative priority used by the Best Master Clock Algorithm (BMCA) of the PTP protocol. Typically controllers, switches, sources will become masters over I/O because they have a better settings.

Grandmaster capable devices may dynamically adjust clock class, clock accuracy, and time source settings. For example, a GPS clock will dynamically change its clock class

and accuracy once it has locked onto satellites. Priority settings may be changed on a particular device to over-ride Best Master Clock selection behavior.

Table 19 – Default PTP clock settings

PTP clock attribute	Controller	I/O device	Switch or router	Primary source
Clock Type(s)	Ordinary (Master / Slave)	Ordinary (Slave Only)	Boundary and/or Transparent	Ordinary (Master Only)
Clock Class	248	255	248	248
Clock Accuracy	Unknown	Unknown	Unknown	Unknown
Clock Variance	Device specific	Device specific	Device specific	Device specific
Time Source	Oscillator	0scillator	Oscillator	Oscillator
Priority1	128	128 (not settable)	128	128
Priority2	128	128 (not settable)	128	128

m) TTL – Time to Live (CP 2/2 only)

By default, the TTL value in the UDP/IPV4 packet shall be set to 1 on CP 2/2 networks. This is required to prevent PTP packets from traversing routers which could significantly increase packet jitter. Where required, PTP time can be made to cross subnets by using routers with IEC 61588 boundary clocks.

n) Hand_Set clock quality management

Some CPF 2 grandmaster clocks allow the time to be set directly by the user or through a user administered mechanism. For example, a device may provide a configuration option to set the System Time from a personal computer (PC). This behavior is characterized as a Hand_Set clock. The process of hand setting a clock should improve the quality of the clock, since the new time will presumably be a more accurate reference. A clock may for example powerup with an unknown time. Once set the time is known and to within a specified tolerance of UTC time.

Table 20 defines PTP settings for a clock that supports Hand_Set behavior. These settings are defined such that all clocks implementing Hand_Set behavior will interoperate. The clock class and accuracy combinations are identified along with the time source. The time source is an information only identifier and does not factor in the choice of best master.

Table 20 – Hand_Set clock quality management

Clock condition	Clock class	Clock accuracy	Time source
Immediately after hand setting.	Degradated Reference (B) (187)	Within 10 s (seconds) (0x30)	Hand_Set (0x60)
Powerup out of the box	Default (248)	Unknown (0xFE)	Oscillator (0xA0)
NOTE A degraded reference B (187) is chosen here for clock class instead of a primary reference (6) or Holdover reference (7) class because the Degraded reference B allows the clock to be a slave when a better master is chosen. Specifically this means the Hand_Set clock will synchronize its clock to the current grandmaster clock rather than going into a PASSIVE master state. Additionally, no consideration is given to aging the clock quality over time – e.g. decaying to a lower quality clock as the accuracy drifts over time, but such behavior is acceptable.			

o) Support for Device Level Ring and path reconfiguration

Devices that support ring topologies and require high synchronization accuracy, such as CIP Motion, shall support automatic path re-measurement whenever there is a change in the ring topology. These devices will either directly monitor the network for a change in the network topology or indirectly be notified by other nodes that are monitoring the network.

Both DLR gateway and DLR end nodes shall implement the CIP Sync Path Reconfiguration Signalling message (see 6.2.1.2.6.5, p)).

NOTE See IEC 61158-4-2, Clause 10, for more information regarding Device Level Ring (DLR) and CIP Sync support.

In DLR, a topology change requires that CIP Sync slave devices re-compute the path delay measurement to the master when the path changes. For example, a break in the ring will cause the path and the corresponding path measurement between the master and the slave to change. It is recommended that this computation be completed as soon as possible, within the next few sync cycles, after a network reconfiguration is detected.

A PTP slave device detects that the path has changed by one of two mechanisms. A slave device in the ring will detect that the ring has broken as indicated by the ring status of the ring protocol. A slave device outside the ring will detect the path has changed when it receives a CIP Sync Path Reconfiguration Signalling message from the bridge node.

DLR gateway devices that bridge between a ring and external network should generate the Signalling message on the non-ring bridged network when a ring break is detected and the master resides within the ring.

Both of these scenarios are illustrated in Figure 11. The path between master M and slave S1 before the ring break is the path labelled 1. After the ring break the path is 2 + 3 + 4. Node S1 will detect the ring break and re-measure the path. The path between master M and slave S2 before the ring break is 1 + 2 + 3 + 5. After the ring break the path is 4 + 5. The DLR gateway device between paths 4 and 5 will detect the ring break and signal node S2 to re-measure the path.

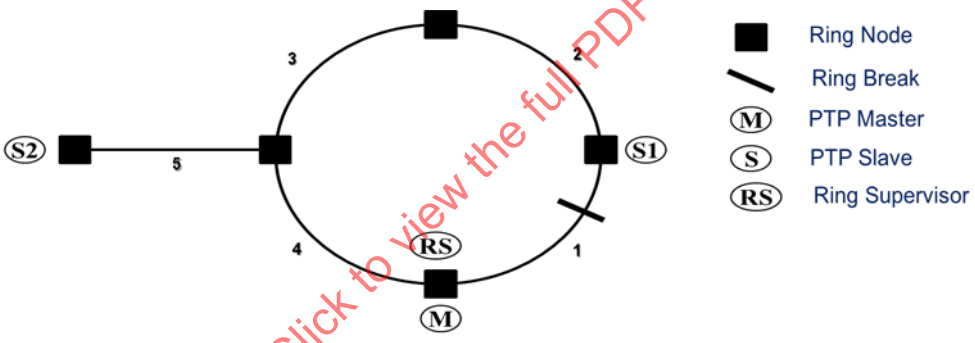


Figure 11 – Path Reconfiguration in a ring topology

p) Path Reconfiguration Signalling message

The Path Reconfiguration Signalling message is issued by a device when a network reconfiguration is detected. This message is a CIP Sync profile specific Signalling message. Ring and fault tolerant time masters and other network devices will issue this message when a network reconfiguration has been detected. When this message is received by a node, it should recalculate its mean path delay using the delay request-response mechanism. Details of the Signalling message are defined in Table 21 and IEC 61588.

Table 21 – Path Reconfiguration Signalling message

Signalling TLV	Field description	Field value
TLV Type	Organization extension	03
Organization Id	CIP Sync Id	00 21 6c
TLV Subtype	Path reconfiguration	01

q) Clock model

CPF2 time synchronization defines an offset clock model to address the requirements for industrial control applications.

NOTE This model is needed because, even though the IEC 61588:2009 PTP protocol defines a mechanism for distributing and synchronizing time, it does not define a mechanism to compensate for step changes in time that could occur at the grandmaster clock source.

Step changes in time may occur at the grandmaster clock source due to one or more of the following conditions.

- The user adjusts the master clock whose type is Hand_Set.
- A master with a more accurate clock becomes available (new grandmaster). This may occur during system startup or after the system has been running for some time.
- The time master is temporarily disconnected from the slave clock and then re-connected. In this situation, given any discrepancy in time between the master and the slave, a step change will occur.

Those applications requiring a stable clock in the presence of step changes in time should implement their PTP clock as a local clock, synchronized to the PTP master frequency, but not to the PTP master's absolute time value.

In this model, the PTP protocol is used to discipline the local clock so that it ticks at the same rate as the master. The device then maintains an offset between the local clock time and system time. A small delta change in time will cause the device to make a small adjustment to the offset and to continue to "tune" its clock. A large change in time will cause the device to update its offset but not "tune" its clock.

Cyclic events may then be scheduled based on the local clock and will not be affected by large step changes in time — provided that the local clock remains synchronized to the PTP master clock frequency.

Figure 12 shows the relationship between the Local Clock, System Time Offset, and the System Time for a PTP Time Slave. The "system to local clock offset" is a memory variable maintained by the PTP clock implementation to make offset adjustments. A leap second value is added to the offset to give System Time in terms of the proper epoch.

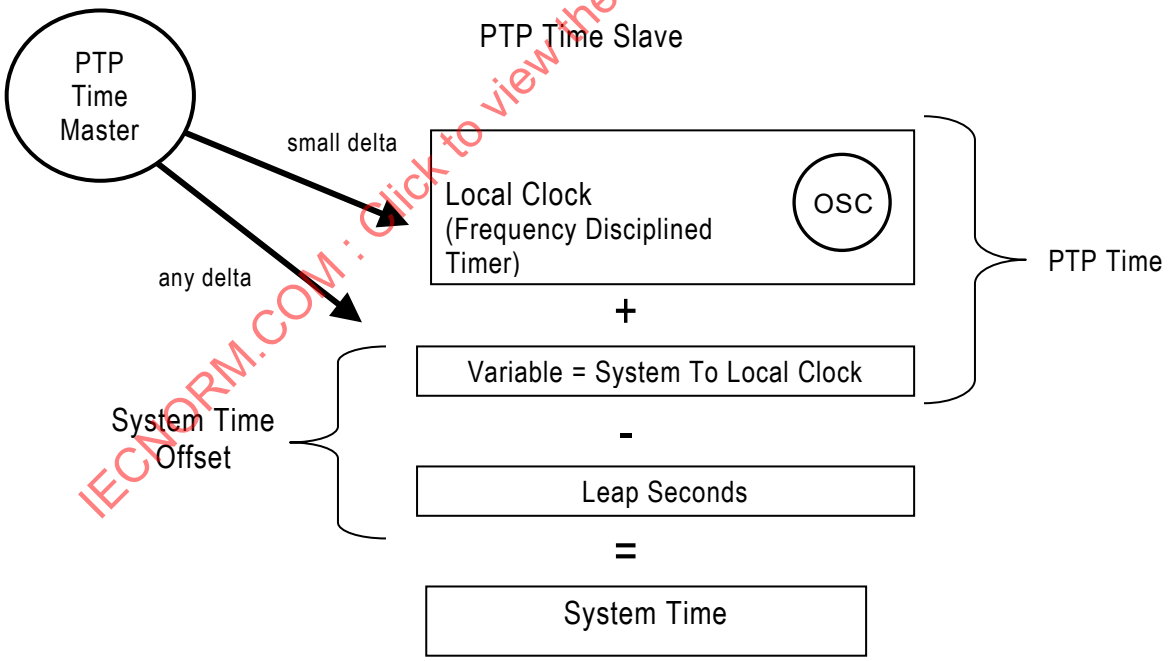


Figure 12 – CPF2 time synchronization offset clock model

Figure 13 shows a system of devices each implementing the CPF2 time synchronization Offset Clock Model. Each device is a part of a network of devices that share the same concept of System Time. Each device also has a Local Clock value based on a frequency disciplined timer and related to System Time via an offset value (System Time Offset). Such a system allows each device to maintain a local time independent from all other

devices, but share a common notion of system time. As such, System Time may change without requiring changes to the local clock.

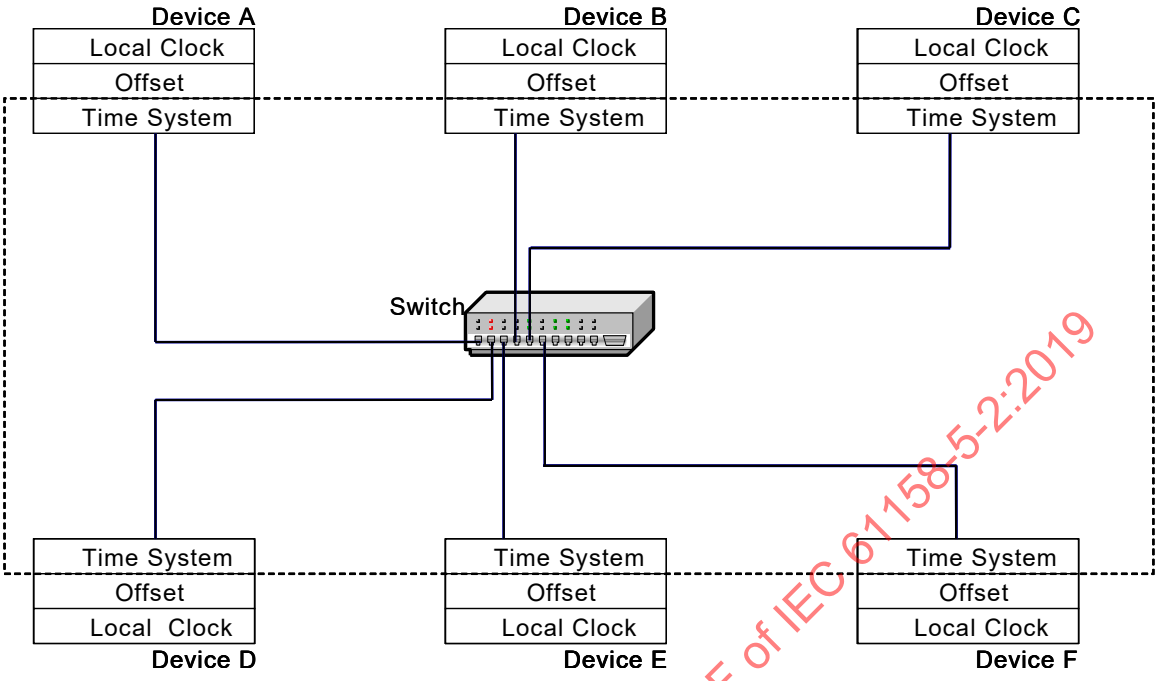


Figure 13 – CPF2 time synchronization system with offset clock model

r) System time compensation

An application creates a timestamp by reading the clock that has been synchronized by PTP and making any required adjustments to the value, for example, converting it to System Time. Additional adjustments may be required if a step has occurred in System Time.

CPF2 time synchronization defines a mechanism to maintain a consistent set of timestamps in the presence of step changes to System Time. A step change in System Time effectively causes a shift in the time base of the system. Any step changes to the grandmaster clock time shall propagate through the system. Since, all nodes in the system will not see the step change at the same instance of time, a timestamp taken on one node may be inconsistent with a timestamp on another node. Or a timestamp taken at one instance in time may be inconsistent with a timestamp taken later in time on the same node. A compensation algorithm is needed to make timestamps consistent with each other before they are used in computations.

Two possible algorithms are described in the following sections.

The decision to compensate timestamps is made based on the requirements of the application.

s) Step compensation algorithms

An offset value is maintained with each timestamp. This offset value is the System Time Offset at the time the timestamp is captured. The offset can be used to provide an indication of when a step occurs in System Time. If the timestamp is transmitted across the network from one node to a second node the offset is sent along with the timestamp. If the two timestamps are captured within the same node, the offsets are stored along with the timestamps.

The algorithms transform a timestamp from one time base to a second time base if a time base change occurs between the times the two timestamps are captured. When the two timestamps are compared they will reference the same time base. These algorithms can equally be applied to a set of timestamps.

t) Compensation algorithm for timestamps within a single node

This algorithm is defined to allow a node to adjust the value of one or more timestamps that have been captured on the local node.

The algorithm is stated as follows:

$$Timestamp_C = Timestamp_0 + Offset_1 – Offset_0$$

where:

$Timestamp_C$ is the compensated timestamp;

$Timestamp_0$ is the timestamp to be compensated;

$Offset_1$ is the offset associated with the timestamp at the target time base;

$Offset_0$ is the offset for the timestamp to be compensated.

u) Compensation algorithm for timestamps between nodes

This algorithm is defined to allow a node to adjust the value of a received timestamp from a remote source node so that it can be compared to a timestamp on its own node, the destination node.

The destination node shall be able to detect a step change to the System Time on the remote node or on its own node and make the appropriate adjustment.

Two conditions are possible:

- The source device has seen a step change in time but the destination device has not, or;
- The destination device has seen a step change in time but the source device has not.

The step change is detected by a change in the SystemTimeOffset by either the source or destination. The source offset is sent to the destination device along with the timestamp. The destination device compares the offset received to the previously received offset to determine if a step change has occurred and adjusts the received timestamp value accordingly.

The algorithm is stated as follows:

$$TimestampCompensated = Timestamp_{Received} + ((Dest_{Offset} – Dest_{LastOffset}) – (Source_{Offset} – Source_{LastOffset}))$$

where

$Timestamp_{Received}$ is received timestamp;

$Dest_{Offset}$ is the current value of the local clock time offset at the destination;

$Dest_{LastOffset}$ is the previous value of the local clock time offset at the destination;

$Source_{Offset}$ is the received value of the local clock time offset from the source;

$Source_{LastOffset}$ is the previous value of the local clock time offset from the source.

Last offsets are updated when:

$$|(Dest_{Offset} – Dest_{LastOffset}) – (Source_{Offset} – Source_{LastOffset})| <= SyncBoundsLimit$$

where

SyncBoundsLimit is a relatively small number that defines that synchronization bounds for the application.

v) Group startup sequence

CPF2 time synchronization defines a group startup sequence and a group synchronizing service. The group startup sequence allows a group of devices to guarantee that their clocks have all been synchronized to the PTP master reference clock before starting an application that depends on System Time. The application defines the specific requirements needed for the group to be considered synchronized.

Each application (object) that wishes to synchronize to a group shall implement the Group_Sync service.

An application may send additional parameters as part of the Group_Sync service. For example, it may exchange the SystemTimeOffset attribute to facilitate timestamp compensation, or a period and phase to initiate a synchronized periodic event, or one or more bounding parameters to specify a target synchronization requirement.

The service returns true if the device is synchronized to the PTP Time master otherwise it returns false.

An example startup sequence for a group of applications that need to synchronize is illustrated in Figure 14.

Once the controller itself is synchronized to the master time clock, it initiates a series of Group_Sync messages to each of the devices also part of the same group. Each device returns a response indicating whether it is also synchronized with the PTP master reference clock. If the controller and all devices are synchronized then the group is synchronized. Otherwise, the master application repeats the synchronizing sequence or takes some fault action.

The Sync'd status of each node is determined by the requirements of the application and can be contingent on additional synchronization parameters.

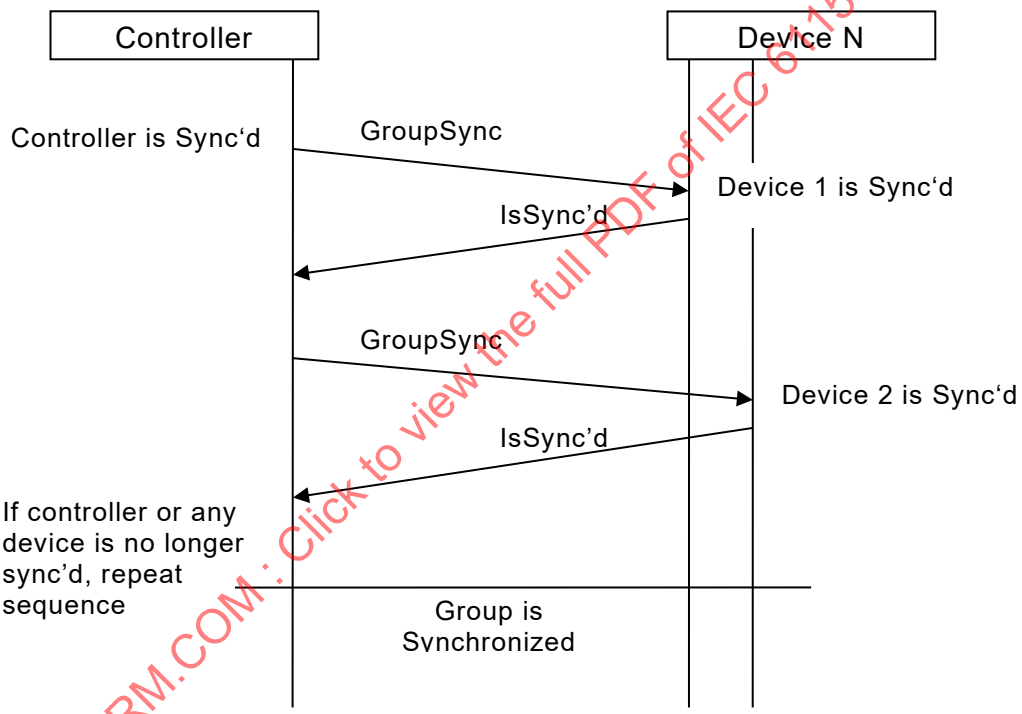


Figure 14 – CPF2 time synchronization group startup sequence

6.2.1.2.7 Parameter formal model

6.2.1.2.7.1 Class definition

Use of the Parameter object provides a known, public interface to a device's configuration data. In addition, this object also provides all the information necessary to define and describe each of a device's individual configuration parameters.

This object allows a device to fully identify a configurable parameter by supplying a full description of the parameter, including minimum and maximum values and a human–readable text string describing the parameter.

There shall be one instance of this object class for each of the device's configurable parameters. The instances shall start at instance one and increment by one with no gaps in the instances.

Each instance is linked to one of the configurable parameters, which is typically (but is not required to be) an attribute of one of the device's other objects. Changing the Parameter Value attribute of a Parameter object causes a corresponding change in the attribute value indicated by the Link Path attribute (the attribute value being pointed to by the Parameter object).

A Parameter Object Stub is a shorthand version of a Parameter object. The Stub stores only the configuration data value, providing only a standard access point for the parameter.

(*) in front of an attribute or a service means that this attribute/service is either mandatory or optional, based on some constraints defined in the attribute/service description.

FAL ASE: **FAL Management ASE**
CLASS: **Parameter_Object**
CLASS ID: **15**
PARENT CLASS: **Base_Object**
ACCESS ATTRIBUTES:

1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):

1	(o)	Attribute:	Revision = 1
2	(o)	Attribute:	Max Instance
8	(m)	Attribute:	Parameter Class Descriptor
9	(m)	Attribute:	Configuration Assembly Instance
10	(o)	Attribute:	Native Language

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):

1	(m)	Attribute:	Parameter Value
2	(m)	Attribute:	Link path size
3	(m)	Attribute:	Link path
4	(m)	Attribute:	Descriptor
5	(m)	Attribute:	Data Type
6	(m)	Attribute:	Data Size
7	(*)	Attribute:	Parameter Name String
8	(*)	Attribute:	Units String
9	(*)	Attribute:	Help String
10	(*)	Attribute:	Minimum Value
11	(*)	Attribute:	Maximum Value
12	(*)	Attribute:	Default Value
13	(*)	Attribute:	Scaling Multiplier
14	(*)	Attribute:	Scaling Divisor
15	(*)	Attribute:	Scaling Base
16	(*)	Attribute:	Scaling Offset
17	(*)	Attribute:	Multiplier Link
18	(*)	Attribute:	Divisor Link
19	(*)	Attribute:	Base Link
20	(*)	Attribute:	Offset Link
21	(*)	Attribute:	Decimal Precision
22	(o)	Attribute:	International Parameter Name
23	(o)	Attribute:	International Engineering Units
24	(o)	Attribute:	International Help String

SYSTEM MANAGEMENT SERVICES:

1	(o)	Mgt Service:	Get_Attributes_All
5	(o)	Mgt Service:	Reset
13	(o)	Mgt Service:	Apply_Attributes
14	(m)	Mgt Service:	Get_Attribute_Single
16	(o)	Mgt Service:	Set_Attribute_Single
21	(*)	Mgt Service:	Restore
22	(*)	Mgt Service:	Save

OBJECT MANAGEMENT SERVICES:

1	(*)	Ops Service:	Get_Attributes_All
13	(o)	Ops Service:	Apply_Attributes
14	(m)	Ops Service:	Get_Attribute_Single
16	(m)	Ops Service:	Set_Attribute_Single
21	(*)	Ops Service:	Restore
22	(*)	Ops Service:	Save
24	(*)	Ops Service:	Get_Member

OBJECT SPECIFIC SERVICES:

75	(o)	Ops Service:	Get_Enum_String

6.2.1.2.7.2 System management attributes (class attributes)

Revision

The Revision level of this object.

This instance attribute has an access rule of Get.

Max Instance

Maximum instance number of an object currently created in this class level of the device.

This instance attribute has an access rule of Get.

Parameter Class Descriptor

Bits that describe parameters.

This instance attribute has an access rule of Get.

Configuration Assembly Instance

Instance number of the configuration assembly.

This instance attribute has an access rule of Get.

Native Language

Language ID for all character array accesses.

If the Active Language attribute of the Identity object (see 6.2.1.2.2.3) is supported, then this attribute shall not be supported.

This instance attribute has an access rule of Get/Set.

6.2.1.2.7.3 Object management attributes

Several attributes are mandatory, optional or not allowed, based on one of the three following constraints. The applicable constraint is referred to in the attribute service description. They all depend whether bit 1 in the Parameter Class Descriptor (class attribute 8) is set (supports full attributes).

Constraint (1): If bit is set, this attribute is mandatory, else it is not allowed.

Constraint (2): If bit is set, this attribute is mandatory optional or not allowed, depending on the parameter data type, as specified in IEC 61158-6-2, 4.1.8.7.1.2, else it is not allowed.

Constraint (3): If bit 1 is set, this attribute is optional, else it is not allowed.

Parameter Value

Actual value of parameter. It can be read from or written to. Scaling shall not be performed to this attribute by the Parameter object. Scaling, if enabled, shall be handled by the configuration tool.

This volatile instance attribute has an access rule of Get/Set. Set is mandatory if the Read Only Parameter bit (4) in the Descriptor attribute (4) is FALSE (see IEC 61158-6-2, 4.1.8.7.1.2). This attribute is used in the Stub and Full Parameter object.

Link path size

Size of link path.

This non-volatile instance attribute has an access rule of Get/Set. This attribute is used in the Stub and Full Parameter object.

Link path

CIP path to the object from where this parameter's value is retrieved.

This non-volatile instance attribute has an access rule of Get/Set. This attribute is used in the Stub and Full Parameter object.

Descriptor

Description of parameter.

This non-volatile instance attribute has an access rule of Get. This attribute is used in the Stub and Full Parameter object.

Data Type

Data type code.

This non-volatile instance attribute has an access rule of Get. This attribute is used in the Stub and Full Parameter object.

Data Size

Number of octets in Parameter Value.

This non-volatile instance attribute has an access rule of Get. This attribute is used in the Stub and Full Parameter object.

Parameter Name String

A human readable string representing the parameter name.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Units String

Engineering Unit String.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Help String

Help String.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Minimum Value

The minimum value to which the parameter can be set.

Constraint (2) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Maximum Value

The maximum value to which the parameter can be set.

Constraint (2) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Default Value

The actual value the parameter should be set to when the user wants the default for the parameter.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Multiplier

Multiplier for Scaling Factor.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Divisor

Divisor for Scaling Formula.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Base

Base for Scaling Formula.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Offset

Offset for Scaling Formula.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Multiplier Link

Parameter Instance of Multiplier source.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Divisor Link

Parameter Instance of Divisor source.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Base Link

Parameter Instance of Base source.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Offset Link

Parameter Instance of Offset source.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Decimal Precision

Specifies number of decimal places to use when displaying the scaled engineering value. Also used to determine actual increment value so that incrementing a value causes a change in scaled engineering value to this precision.

Constraint (1) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

International Parameter Name

Human readable string representing the parameter name.

Constraint (3) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

International Engineering Units

Engineering units associated with the parameter.

Constraint (3) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

International Help String

Help String.

Constraint (3) applies. This non-volatile instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

6.2.1.2.7.4 System management (class level) and object management (instance level) services

Get_Attributes_All

At both the instance level (object management) and the class lever (system management), this service returns a list of parameter values from the object.

Reset

At the class level (system management), the Reset service all parameters to the factory default.

Apply_Attributes

At both the instance level (object management) and the class level (system management), this service causes parameter values whose use is pending to become actively used.

Get_Attribute_Single

At both the instance level (object management) and the class level (system management), this service returns the contents of the specified attribute.

Set_Attribute_Single

At both the instance level (object management) and the class level (system management), this service modifies the contents of the specified attribute.

Restore

At the instance level (object management), this service restores parameter instance values from non-volatile storage. At the class level (system management), this service restores all parameter values from non-volatile storage.

Save

At the instance level (object management), this service saves parameter instance values to non-volatile storage. At the class level (system management), this service saves all parameter values to non-volatile storage.

Get_Member

At the instance level (object management), this service returns the content of a selected member of an attribute.

6.2.1.2.7.5 Object specific services

Get_Enum_String

This service is used to read enumerated strings from a Parameter instance.

6.2.1.2.7.6 Parameter state machines

Table 22 is the state event matrix for the Parameter object. The behavior of the Parameter object is illustrated in Figure 15.

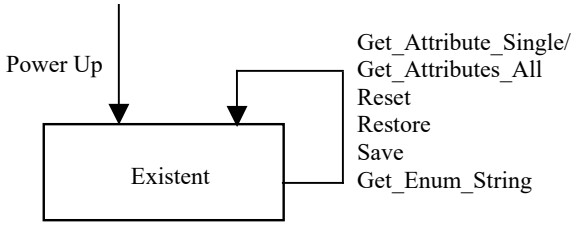


Figure 15 – Parameter object state transition diagram

Table 22 – Parameter object state event matrix

Event	State
	Existent
Get_Attribute_Single.	Validates/service the request and returns the appropriate response
Set_Attribute_Single	
Get_Attributes_All	
Reset	
Restore	
Save	
Get_Enum_String	

6.2.1.3 FAL management model ASE service specification

6.2.1.3.1 Supported services

Subclause 6.2.1.3 contains the definition of services that are unique to this ASE. The common services defined for this ASE are:

Get_Attributes_All
Set_Attributes_All
Get_Attribute_List
Set_Attribute_List
Reset
Start
Stop
Create
Delete
Multiple_Service_Packet
Apply_Attributes
Get_Attribute_Single
Set_Attribute_Single
Find_Next_Object_Instance
Restore
Save
NOP
Get_Member
Set_Member

Insert_Member

Remove_Member

Group_Sync

The object specific services defined for this ASE are:

Add_AckData_Path (Acknowledge Handler object)

Remove_AckData_Path (Acknowledge Handler object)

Get_Enum_String (Parameter object)

Symbolic_Translation (Message Router object)

Flash_LEDs (Identity object)

6.2.1.3.2 FAL service common parameters

The following service parameters are common to all FAL services.

AREP

This parameter contains sufficient information to locally identify the entity to be used to convey the service: this entity could be local or the UCMM or a transport connection (AR). This parameter may use a dedicated identifier associated with a local entity, the identifier of a UCMM transaction record previously created, or the transport identifier returned by the connection establishment process and associated with the selected AR. The confirmation primitive should use the same value as the one sent in the corresponding request primitive.

Receiver/server local ID (id)

This parameter is locally generated by the Message Router ASE of the responding node, and identifies locally this invocation of the service. It is used to associate service responses with indications. Therefore, no two outstanding service invocations may be identified by the same ID (id) value, and the AL-service user shall return in a response primitive the value that it received in the prior associated indication primitive.

Path

In the request, this parameter specifies the FAL APO or FAL APO element that is the destination of the service request. The path shall contain multiple segments which can indicate the network route to the node containing the FAL APO (in the case of multiple links), the identification of the APO element within the target node (Routing Path), and optional additional information for the target APO (Additional Path).

In the indication, this parameter shall contain only those segments beyond the logical class segment from the service request, i.e. the optional additional information for the target APO (AdditionalPath).

See IEC 61158-6-2 for more details on the Path structure and contents.

Port ID

This parameter indicates on which port of the device the service indication arrived.

Service status

This parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements.

Status code (mandatory)

This parameter indicates whether the service was successfully processed or not. If an error occurred, it indicates the type of error. For some errors, further identification of the error may be provided in the Extended Status parameter below. Available status codes are listed in 6.2.1.3.3.

Extended status (conditional)

This conditional parameter further identifies the error encountered when processing the request specific to the object being accessed. Actual usage and definition of this Extended status is dependant on the object class or service: each object class defines its own extended status values and value ranges (including the vendor specific ones). For a given object class, extended status are unique to each status code. All extended status values are reserved unless otherwise indicated within the object definition. When used, the values submitted in the response primitive are delivered unchanged in the confirmation primitive.

These codes may also be used for other purposes such as event logging or in device hearbeat messages.

6.2.1.3.3 FAL service status codes

Table 23 specifies the range of possible status codes.

Table 23 – Status codes

Name	Description and meaning of status code
Success	Service was successfully performed by the object specified
Connection failure	A connection related service failed along the connection path
Resource unavailable	Resources needed for the object to perform the requested service were unavailable
Invalid parameter value	See status code 0x20, which is the preferred value to use for this condition
Path segment error	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered
Path destination unknown	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered
Partial transfer	Only part of the expected data was transferred
Connection lost	The messaging connection was lost
Service not supported	The requested service was not implemented or was not defined for this class or object instance
Invalid attribute value	Invalid attribute data detected
Attribute list error	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non zero status
Already in requested mode/state	The object is already in the mode/state being requested by the service
Object state conflict	The object cannot perform the requested service in its current mode/state
Object already exists	The requested instance of object to be created already exists
Attribute not settable	A request to modify a non-modifiable attribute was received
Privilege violation	A permission/privilege check failed
Device state conflict	The device's current mode/state prohibits the execution of the requested service
Response data too large	The data to be transmitted in the response buffer is larger than the allocated response buffer
Fragmentation of a primitive value	The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type
Not enough data	The service did not supply enough data to perform the specified operation
Attribute not supported	The attribute specified in the request is not supported
Too much data	The service supplied more data than was expected
Object does not exist	The object specified does not exist in the device
Service fragmentation sequence not in progress	The fragmentation sequence for this service is not currently active for this data
No stored attribute data	The attribute data of this object was not saved prior to the requested service
Store operation failure	The attribute data of this object was not saved due to a failure during the attempt.
Routing failure, request packet too large	The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service
Routing failure, response packet too large	The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service
Missing attribute list entry data	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior
Invalid attribute value list	The service is returning the list of attributes supplied with status information for those attributes that were invalid
Embedded service error	An embedded service resulted in an error

Name	Description and meaning of status code
Vendor specific error	A vendor specific error has been encountered. The extended status code field of the error response defines the particular error encountered. Use of this general status code should only be performed when none of the status codes presented in this table or within an object class definition accurately reflect the error
Invalid parameter	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an application object specification
Write once value or medium already written	An attempt was made to write to a write once medium (e.g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established
Invalid Response Received	An invalid response is received (e.g. response service code does not match the request service code, or response message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid responses
Buffer overflow	The message received is larger than the receiving buffer can handle. The entire message was discarded
Message format error	The format of the received message is not supported by the server
Key Failure in path	The Key Segment which was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
Path Size Invalid	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
Unexpected attribute in list	An attempt was made to set an attribute that cannot be set at this time.
Invalid Member ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute.
Member not settable	A request to modify a non-modifiable member was received.
Group 2 only server general failure	This status code may only be reported by CP 2/3 Group 2 Only servers with 4 koctets or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable
Unknown Type 15 error	A Type 2 to Type 15 translator received an unknown Type 15 Exception Code
Attribute not gettable	A request to read a non-readable attribute was received
Instance not deletable	The requested object instance cannot be deleted
Service not supported for specified path	The object supports the service, but not for the designated application path (e.g. attribute). This code shall not be used for any Set service
Reserved for object class specific errors	This range of status codes shall be used to indicate object class specific errors. Use of this range should only be performed when none of the status codes presented in this table accurately reflect the error that was encountered

All extended status values are reserved unless otherwise indicated within the object definition. Specific values are specified when appropriate in IEC 61158-6-2.

6.2.1.3.4 Get_Attributes_All

6.2.1.3.4.1 Service overview

The Get_Attributes_All service returns the contents of all attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

6.2.1.3.4.2 Service primitives

The service parameters for this service are shown in Table 24.

Table 24 – Get_Attributes_All service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request. There are no specific parameters for this service.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute data

This parameter returns a stream of information containing the values of each attribute that the class/object defines for the response. APO classes which support this service shall define the format of this parameter. No fragmentation is allowed.

If supported, the structure of the Attribute Data information shall adhere to the Get_Attributes_All response structure as defined by the object class specification. The object class specification shall provide a detailed definition of the format of the data to be sent in the class and instance response messages.

Default values shall be inserted for all unsupported attributes present in the response data array.

For attributes that specify a count of array elements in another attribute, a zero count and no array elements shall be inserted if the attribute pair is not supported.

For attributes that specify a count followed by an array of entries, a zero count shall be inserted if the attribute is not supported.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Response data too large

6.2.1.3.4.3 Service procedure

A device is only required to include data in the response data array up to the last implemented attribute.

If the length of data in the response is less than documented in the object definition and the last part of the response data does not include a complete attribute, the response data is in error and the client shall discard it.

If the length of data in the response is less than documented in the object definition, the client assumes default values for the missing attributes.

If the length of data in the response is greater than the expected amount, the client ignores any data in excess of what was expected.

6.2.1.3.5 Set_Attributes_All

6.2.1.3.5.1 Service overview

The Set_Attributes_All service modifies the contents of the attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

6.2.1.3.5.2 Service primitives

The service parameters for this service are shown in Table 25.

Table 25 – Set_Attributes_All service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Data	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Attribute data

This parameter specifies a stream of information containing the values of each attribute that the class/object defines for the request. APO classes which support this service shall define the format of this parameter.

If supported, the structure of the Attribute Data information in the service request shall adhere to the Set_Attributes_All request structure as defined by the object class specification. The object class specification shall provide a detailed definition of the format of the data to be sent in the request message.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Device state conflict

6.2.1.3.5.3 Service procedure

If the ability to modify an attribute changes based on the state of the object, the object specification shall specify when its attributes may be written.

EXAMPLE This service could only be supported in a state where all attributes are modifiable.

Attributes shall be modified only if all attribute specific value verifications (e.g. range checks) are successful. The first attribute failing verification will be specified in the Additional Code parameter of the error response message. If any error is detected, the Set_Attributes_All response shall return an appropriate error response message. If all verification checks pass, then the attributes shall be modified and a Set_Attributes_All success response shall be returned.

If the length of data in the service is less than documented in the object definition and the last part of the request data does not include a complete attribute, the “Not enough data” (0x13) error response is returned.

If the length of data in the service data field is less than documented in the object definition, this indicates the attributes represented by the missing data are not supported by the client. Missing data shall result in no modification to the corresponding attributes.

If the length of data in the request is greater than the expected amount, the “Too much data” (0x15) error response is returned.

If a device’s implementation of an object does not support one or more of the optional attributes contained in that object’s Set_Attributes_All request format, then it shall either:

- omit support for the service (if the object definition allows);
- reject the request with an "Attribute Not Supported" (0x14) error; or
- accept the request provided the value(s) contained in the request for the unimplemented attribute(s) are the default values defined for those attribute(s). If any of the values for the unimplemented attribute(s) are not the default value defined for those attribute(s), the request shall be rejected. An “Invalid Attribute Value” (0x09) error code is recommended for this case.

6.2.1.3.6 Get_Attribute_List

6.2.1.3.6.1 Service overview

The Get_Attribute_List service returns the contents of the selected gettable attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

6.2.1.3.6.2 Service primitives

The service parameters for this service are shown in Table 26.

Table 26 – Get_Attribute_List service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Count	M	M(=)		
Attribute List	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Count			M	M(=)
Attribute Data			M	M(=)
Attribute Identifier			M	M(=)
Attribute Status			M	M(=)
Attribute Value			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Attribute count

This parameter indicates the number of attribute identifiers in the Attribute List parameter below.

Attribute list

This parameter contains the list of the top-level attribute identifiers for the attributes that are being requested from the specified class or object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute count

This parameter indicates the number of attributes actually returned in the Attribute Data parameter below. This count can be different if the requested data does not fit entirely in the response buffer (see service procedure in 6.2.1.3.6.3).

Attribute data

This parameter returns a stream of information containing all or part of the requested attributes. Attributes shall be retrieved and packed in the sequence specified in the request.

Attribute identifier

This parameter contains the attribute identifier corresponding to the value being returned in Attribute Value.

Attribute status

This parameter indicates the individual status information for the requested attribute. It can either mirror one of the common service status codes, or be specific to this object/class attribute.

Attribute value

This parameter returns a stream of information containing all data for the requested attribute. Individual attribute data shall fit into the response buffer in its entirety (without fragmentation of individual attributes). The service shall access as many attributes as can fit into the response buffer.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Attribute list error

6.2.1.3.6.3 Service procedure

The attributes shall be specified using a list of attribute identifiers, which shall be supplied as a service parameter. If there is not enough space for an attribute's data in the response buffer, service processing shall terminate and the Attribute Count parameter shall be set to the number of attributes actually packed in the buffer. The client application (the requester), shall verify the value of the Attribute Count parameter in the response.

NOTE If necessary, the client application can submit another Get_Attribute_List service request for the attribute data remaining from its original list.

6.2.1.3.7 Set_Attribute_List

6.2.1.3.7.1 Service overview

The Set_Attribute_List service updates the contents of the selected attributes of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

6.2.1.3.7.2 Service primitives

The service parameters for this service are shown in Table 27.

Table 27 – Set_Attribute_List service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Count	M	M(=)		
Attribute Data	M	M(=)		
Attribute Identifier	M	M(=)		
Attribute Value	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Count			M	M(=)
Attribute Status List			M	M(=)
Attribute Identifier			M	M(=)
Attribute Status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Attribute count

This parameter indicates the number of attributes to be updated, and listed in the Attribute Data parameter below.

Attribute data

This parameter contains a stream of information containing the actual list of attributes to be updated and the corresponding update values.

Attribute identifier

This parameter specifies the attribute identifier corresponding to the attribute to be updated with the contents of Attribute Value. The client is unable to specify sub-elements of an attribute.

Attribute value

This parameter contains a stream of information containing all data for the target attribute. The data for an individual attribute shall be placed into the request buffer in its entirety (without fragmentation).

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute count

This parameter indicates the number of attribute status returned in the Attribute Status List parameter below.

Attribute status list

This parameter returns a stream of information containing status information for each attribute listed in the service request.

Attribute identifier

This parameter contains the attribute identifier corresponding to the status being returned in Attribute Status.

Attribute status

This parameter indicates the individual status information for the specified attribute. It can either mirror one of the common service status codes, or be specific to this object/class attribute.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Attribute list error

Device state conflict

Missing attribute list entry data

6.2.1.3.7.3 Service procedure

The service shall be terminated if the response buffer is unable to accept the status of the next attribute.

Individual set operations shall not be interdependent (the request shall be treated as a series of independent set requests).

6.2.1.3.8 Reset

6.2.1.3.8.1 Service overview

The Reset service invokes the Reset service of the specified APO object class or instance.

NOTE Typically this would cause a transition to a default state or mode.

6.2.1.3.8.2 Service primitives

The service parameters for this service are shown in Table 28.

Table 28 – Reset service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Invalid parameter value

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Object state conflict

Device state conflict

6.2.1.3.8.3 Service procedure

If the execution of the Reset service request would place the object/device in a state that enables it to respond to the requester, then the response shall not be returned until the service has completed. If the execution of the Reset service would place the object/device in a state in which it is not able to respond, the object/device shall respond prior to executing the Reset service.

6.2.1.3.9 Start

6.2.1.3.9.1 Service overview

The Start service invokes the Start service of the specified APO object class or instance.

NOTE Typically this would place an object into a running state/mode.

6.2.1.3.9.2 Service primitives

The service parameters for this service are shown in Table 29.

Table 29 – Start service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Already in requested mode/state

Object state conflict

Device state conflict

6.2.1.3.9.3 Service procedure

If supported by an object class or instance, the effect of this service shall be as documented in that object specification.

6.2.1.3.10 Stop

6.2.1.3.10.1 Service overview

The Stop service invokes the Stop service of the specified APO object class or instance.

NOTE Typically this would place an object into a stopped or idle state/mode.

6.2.1.3.10.2 Service primitives

The service parameters for this service are shown in Table 30.

Table 30 – Stop service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport Identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Already in requested mode/state

Object state conflict

Device state conflict

6.2.1.3.10.3 Service procedure

The effect of this service shall be documented in the individual object specification.

6.2.1.3.11 Create

6.2.1.3.11.1 Service overview

The Create service results in the instantiation of a new object instance within the specified object class.

6.2.1.3.11.2 Service primitives

The service parameters for this service are shown in Table 31.

Table 31 – Create service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Instance ID			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Instance ID

This mandatory parameter indicates the integer value assigned to identify the newly created object instance.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error
Path destination unknown
Connection lost
Service not supported
Invalid attribute value
Already in requested mode/state
Object state conflict
Device state conflict
Missing attribute list entry data
Invalid attribute value list

6.2.1.3.11.3 Service procedure

If this service is supported, the object instance shall be created and initialized in accordance with the object specification.

Any error shall result in the object instance not being created.

6.2.1.3.12 Delete

6.2.1.3.12.1 Service overview

The Delete service deletes an object instance of the specified object class.

6.2.1.3.12.2 Service primitives

The service parameters for this service are shown in Table 32.

Table 32 – Delete service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object Specific Data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Object state conflict

Device state conflict

6.2.1.3.12.3 Service procedure

All resources shall be deallocated and returned to the system.

6.2.1.3.13 Get_Attribute_Single

6.2.1.3.13.1 Service overview

The Get_Attribute_Single service returns the contents of the specified attribute (or subattribute specified by Bit Index, Indirect Bit Index, Array Index, Indirect Array Index, Structure Member or Structure Handle) of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

6.2.1.3.13.2 Service primitives

The service parameters for this service are shown in Table 33.

Table 33 – Get_Attribute_Single service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request. There are no specific parameters for this service, except as noted below.

Path

This parameter includes a path segment specifying the attribute number.

NOTE See IEC 61158-6-2 for a description of path segments.

Some CPF 2 profiles support link specific explicit message formats (i.e. they do not support the Message Router request format) which do not enable the attribute number to be specified in the Path parameter. For those formats, the attribute number is transmitted as an additional parameter.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute data

This parameter contains the requested attribute data.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Connection lost

Service not supported

Invalid attribute value

Attribute not settable

Device state conflict

Object does not exist

6.2.1.3.13.3 Service procedure

No specific service procedure.

6.2.1.3.14 Set_Attribute_Single

6.2.1.3.14.1 Service overview

The Set_Attribute_Single service modifies the contents of the specified attribute (or subattribute specified by Bit Index, Indirect Bit Index, Array Index, Indirect Array Index, Structure Member or Structure Handle) of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

6.2.1.3.14.2 Service primitives

The service parameters for this service are shown in Table 34.

Table 34 – Set_Attribute_Single service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Data	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request. There are no specific parameters for this service.

Path

This parameter includes a path segment specifying the attribute number.

NOTE See IEC 61158-6-2 for a description of path segments.

Attribute data

This parameter specifies the value to which the specified attribute is to be modified. The attribute data shall be validated prior to the modification taking place.

Some CPF 2 profiles support link specific explicit message formats (i.e. they do not support the Message Router request format) which do not enable the attribute number to be specified in the Path parameter. For those formats, the attribute number is transmitted as the fist parameter (octet) of the attribute data.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Attribute not settable

Device state conflict

Object does not exist

6.2.1.3.14.3 Service procedure

No specific service procedure.

6.2.1.3.15 Find_Next_Object_Instance

6.2.1.3.15.1 Service overview

The Find_Next_Object_Instance service shall be supported at the APO object class level only. It causes the specified APO object class to search for and return a list of Instance IDs associated with existing object instances. Existing objects are those that are currently accessible from the link.

6.2.1.3.15.2 Service primitives

The service parameters for this service are shown in Table 35.

Table 35 – Find_Next_Object_Instance service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Receiver/Server Local ID	M	M(=)		
Path size	M	M(=)		
Path	M	M(=)		
Port ID		M		
Maximum Returned Values	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Number of List Members			M	M(=)
Instance ID List			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Path

This parameter includes a path segment specifying an Instance ID, which will be used to determine the starting point for the search.

Maximum returned values

This parameter indicates the maximum number of Instance ID values to be returned in the response message.

Result(+)

This selection type parameter indicates that the service request succeeded.

Number of list members

This parameter contains the actual number of Instance ID's returned in this response message. This number of Instance ID's shall be less than or equal to the maximum specified in the request.

Instance ID list

This parameter contains the list of returned Instance ID's.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Object state conflict

Device state conflict

6.2.1.3.15.3 Service procedure

The object class shall utilize the value specified in the Instance ID of the request message to determine the starting point for the search as described below:

1) If the Instance ID in the request message is zero (0), then the class shall start with the numerically lowest Instance ID;
2) If the Instance ID in the request message is not zero (0), then the class shall start with the next Instance ID whose value is numerically greater than the specified Instance ID.
3) If the Instance ID in the request message is greater than or equal to the numerically highest Instance ID within the class, then the value zero (0) shall be returned as Instance ID.

The class shall return a list of Instance IDs associated with existing objects, beginning at the starting point and continuing in ascending Instance ID value order. It shall return Instance ID value zero (0) to indicate that the end of the list has been reached.

Figure 16 provides a general example of this service.

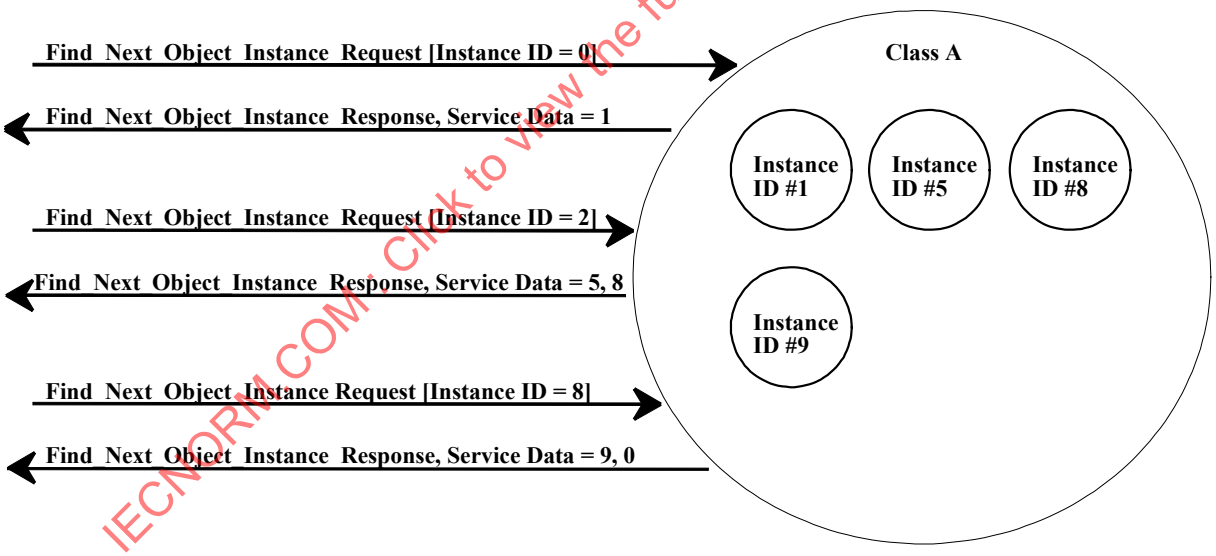


Figure 16 – Example of Find_Next_Object_Instance service

6.2.1.3.16 NOP

6.2.1.3.16.1 Service overview

The NOP service causes the receiving APO object instance to return a *No Operation* response. The receiving APO object instance shall not carry out any other internal action.

NOTE This service can be used to test whether or not a particular object is still present and responding without causing a state change.

6.2.1.3.16.2 Service primitives

The service parameters for this service are shown in Table 36.

Table 36 – NOP service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

6.2.1.3.16.3 Service procedure

If the object instance to which the request is delivered supports the NOP service, then a response whose status indicates success shall be returned. If the object does not support the NOP service, then a response indicating an error was detected shall be returned.

6.2.1.3.17 Apply_Attributes

6.2.1.3.17.1 Service overview

The Apply_Attributes service causes attribute values whose use is pending to become actively used in the specified APO object class or instance.

6.2.1.3.17.2 Service primitives

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

Table 37 – Apply_Attributes service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Object state conflict

Device state conflict

6.2.1.3.17.3 Service procedure

Data for pending attributes shall be verified before using them.

6.2.1.3.18 Save

6.2.1.3.18.1 Service overview

The Save service copies the contents of an APO object class or instance attributes to a location accessible by the Restore service.

6.2.1.3.18.2 Service primitives

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

Table 38 – Save service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error
Path destination unknown
Connection lost
Service not supported
Object state conflict
Device state conflict
Store operation failure

6.2.1.3.18.3 Service procedure

The Save service shall report success only after the copy has been completed and verified.

6.2.1.3.19 Restore

6.2.1.3.19.1 Service overview

The Restore service restores the contents of an APO object class or instance attributes from a storage location accessible by the Save service. Attribute data shall be copied from a storage area to the currently active memory area used by the object class or instance.

6.2.1.3.19.2 Service primitives

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

Table 39 – Restore service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Object state conflict

Device state conflict

No stored attribute data

6.2.1.3.19.3 Service procedure

Attribute data shall be verified before performing the copy from the “storage area” to the “actively used” area.

If the ability to modify an attribute changes based on the state of the object, the object specification shall provide a detailed description of how this service is effected. This service may only be supported in a state where all attributes are modifiable. The object may ignore the data associated with a currently non-modifiable attribute.

Attributes shall be modified only if all attribute specific value verifications (e.g. range checks) are successful. The first attribute failing verification shall be specified in the Extended Status element of the Service Status parameter of the response message.

If any other error is detected, then the response shall be returned with a non-zero status code.

If all verification checks pass, then the attributes shall be modified and a Restore success response shall be returned.

6.2.1.3.20 Get_Member

6.2.1.3.20.1 Service overview

The Get_Member service returns member(s) information from within an attribute.

6.2.1.3.20.2 Service primitives

The service parameters for this service are shown in Table 40.

Table 40 – Get_Member service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Member ID			M	M(=)
Member data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request. There are no specific parameters for this service.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Member ID

This parameter contains the identification of the member which has been serviced.

Member data

This parameter contains member(s) information from the selected attribute.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.20.3 Service procedure

The following list details requirements associated with the Get_Member service.

- The service causes the class/instance to return member(s) at the specified Member ID of an attribute.
- If the Member ID value is greater than that of the last Member ID for the specific attribute, the member with the highest Member ID is returned.
- If the Member ID value is zero, an error response with status “Invalid Member ID” is returned.
- The Get_Attribute_Single service shall be used to get all members.

When the International String Selection member protocol is used for the Get_Member service, the following list details corresponding additional requirements.

- This member protocol causes the class/instance to return an international string from an attribute of data type STRINGI.
- If the object or attribute to which the request is delivered does not support the service/protocol, then an error response with status “Service Not Supported” is returned.
- If the attribute does not contain the requested language, then an error response with status “Invalid Parameter” is returned.
- If the Member ID parameter is zero, then the device shall return the active (default) language. This default language is set by the device and may optionally be changed through the Active Language attribute (Attribute ID 11) within the Identity object.
- A device shall support the Supported Language List attribute (Attribute ID 12) of the Identity object if this member protocol is supported.
- If the requested Member ID is valid but the string for that language is not currently available then an error response with status “Resource Unavailable” shall be returned and no additional error code. The string for the active language (as indicated by the Active Language attribute of the Identity Object) shall always be available.

6.2.1.3.21 Set_Member

6.2.1.3.21.1 Service overview

The Set_Member service sets member(s) information in an attribute.

6.2.1.3.21.2 Service primitives

The service parameters for this service are shown in Table 41.

Table 41 – Set_Member service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Member data	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Member ID			C	C(=)
Member data			U	U(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Member data

This parameter contains member(s) information for the selected attribute.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Member ID

This conditional parameter is mandatory if Member Data is present, else it is optional. If present, it contains the identification of the member which has been serviced.

Member data

This optional parameter, if present, contains member(s) information from the selected attribute.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.21.3 Service procedure

The following list details requirements associated with the Set_Member service.

- The service causes the class/instance to set member(s) at the specified Member ID of an attribute.
- The request is checked, and if valid, the member(s) are set to the new Member Data.
- If the Member ID value is greater than that of the last Member ID for the specific attribute, no action occurs and an error response with status “Invalid Member ID” is returned.
- If multiple members are specified and fewer members exist at the specified Member ID, no action occurs and an error response with status “Invalid Member ID” is returned.
- If the Member ID value is zero, an error response with status “Invalid Member ID” is returned.

6.2.1.3.22 Insert_Member

6.2.1.3.22.1 Service overview

The Insert_Member service inserts member(s) into an attribute.

6.2.1.3.22.2 Service primitives

The service parameters for this service are shown in Table 42.

Table 42 – Insert_Member service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Member data	U	U(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Member ID			M	M(=)
Member data			U	U(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Member data

This optional parameter, if present, contains member(s) information for the selected attribute.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Member ID

This parameter contains the identification of the member which has been serviced.

Member data

This optional parameter, if present, contains member(s) information from the selected attribute.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.22.3 Service procedure

The following list details requirements associated with the Insert_Member service.

- The service causes the class/instance to insert member(s) at the specified Member ID of an attribute. The Member IDs of members at or following the specified Member ID will change.
- The request is checked, and if valid, the member(s) are inserted. If Member Data is not included in the request, the member(s) are set to the class/attribute specific default.
- If an error is detected, then no action is taken.
- If the number of members specified cannot be added to the attribute, then an error response with status “Resource unavailable” is returned.
- If the Member ID value is greater than that of the last Member ID for the specific attribute, the service appends member(s) to the attribute and returns the Member ID where inserted.
- If the Member ID value is zero, an error response with status “Invalid Member ID” is returned.

6.2.1.3.23 Remove_Member

6.2.1.3.23.1 Service overview

The Remove_Member service removes member(s) from an attribute.

6.2.1.3.23.2 Service primitives

The service parameters for this service are shown in Table 43.

Table 43 – Remove_Member service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Member ID			M	M(=)
Member data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request. There are no specific parameters for this service.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Member ID

This parameter contains the identification of the member which has been serviced.

Member data

This parameter contains member(s) information from the selected attribute.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.23.3 Service procedure

The following list details requirements associated with the Remove_Member service.

- The service causes the class/instance to remove member(s) at the specified Member ID of an attribute. The Member IDs of members following the removed member(s) change.
- If an error is detected, then no action is taken.
- If the Member ID value is greater than that of the last Member ID for the specific attribute, the member with the highest Member ID is removed.
- If multiple members are specified and fewer members exist at the specified Member ID, the member(s) that do exist are removed.

6.2.1.3.24 Group_Sync

6.2.1.3.24.1 Service overview

The Group_Sync service verifies that each member of a group is synchronized to System Time.

6.2.1.3.24.2 Service primitives

The service parameters for this service are shown in Table 44.

Table 44 – Group_Sync service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
IsSynchronized			M	M(=)
Object Specific Data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This parameter specifies object class/instance specific parameters. The object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

IsSynchronized

This parameter indicates if the object is synchronized to the PTP Time Master.

Object specific data

This parameter contains object class/instance specific parameters. The object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.24.3 Service procedure

The following list details requirements associated with the Group_Sync service.

- The structure of the information in the request's Service Data Field adheres to the definition of the Group_Sync request for the object or class. Support of this service requires the object and/or class to provide a detailed definition of the format of the data sent in the request message.
- The structure of the information in the response's Service Data Field adheres to the definition of the Group_Sync response for the object or class. Support of this service requires the object and/or class to provide a detailed definition of the format of the data sent in the response message.
- The responder will verify that the application is synchronized according to the object specific requirements. The target return's a 1 in the IsSynchronized attribute of the response Service Data Field if the synchronized status is true and a 0 if the synchronized status is false.

See the Time Sync ASE specification in 6.2.1.2.6 for a more detailed description of the Group_Sync service.

6.2.1.3.25 Add_AckData_Path

6.2.1.3.25.1 Service overview

The Add_AckData_Path adds a path for data with acknowledgment for a connected consumer.

6.2.1.3.25.2 Service primitives

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

Table 45 – Add_AckData_Path service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Connection Instance ID	M	M(=)		
Consumer Path Size	M	M(=)		
Consumer Path	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Connection instance ID

This parameter identifies an active connection instance which is receiving Acks.

Consumer path size

Size of Consumer Path (in octets).

Consumer path

Internal path to the application object consuming data received with acknowledgment (padded format). See IEC 61158-6-2 for the format of padded path.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.25.3 Service procedure

No specific service procedure.

6.2.1.3.26 Remove_AckData_Path

6.2.1.3.26.1 Service overview

The Remove_AckData_Path service removes a path for data with acknowledgement for the given connected consumer.

6.2.1.3.26.2 Service primitives

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

Table 46 – Remove_AckData_Path service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Connection Instance ID	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Connection instance ID

This parameter identifies the active connection instance which is receiving Acks and which path should be removed.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.26.3 Service procedure

No specific service procedure.

6.2.1.3.27 Get_Enum_String

6.2.1.3.27.1 Service overview

The Get_Enum_String service reads enumerated strings from a Parameter instance.

6.2.1.3.27.2 Service primitives

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

Table 47 – Get_Enum_String service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Enumerated String Number	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Enumerated String			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Enumerated String Number

This parameter indicates the number of the enumerated string to retrieve.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Enumerated String

This parameter contains the requested enumerated string.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.27.3 Service procedure

No specific service procedure.

6.2.1.3.28 Symbolic translation

6.2.1.3.28.1 Service overview

The Symbolic translation service provides a translation from a Symbolic Segment path encoding to the equivalent Logical Segment path encoding, if it exists.

6.2.1.3.28.2 Service primitives

The service parameters for this service are shown in Table 48.

Table 48 – Symbolic_Translation service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Symbolic Address	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Logical Address			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Symbolic Address

This parameter contains a Symbolic Segment or ANSI Extended Symbol Segment to be translated.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Logical Address

This parameter contains a Logical Segment and optional Data Segment encoding representing the internal equivalent of the requested Symbolic Address.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.28.3 Service procedure

No specific service procedure.

6.2.1.3.29 Flash_LEDs

6.2.1.3.29.1 Service overview

The Flash_LEDs service instructs the device to either start or stop flashing its Type 2-defined LEDs (used to visually identify it).

6.2.1.3.29.2 Service primitives

The service parameters for this service are shown in Table 49.

Table 49 – Flash_LEDs service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Time	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Time

This parameter indicates the amount of time, in seconds, from when the service was received, that the device flashes its LEDs.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.29.3 Service procedure

The device shall force all of its applicable LEDs to simultaneously display a “red-green-off” sequence, holding each value for 500 ms. This pattern shall be applied to all Type 2-defined LEDs. Type 2-defined LEDs are specific to each CP 2/x. This pattern may also be applied to vendor specific red/green LEDs. For devices that support visual indicators other than red/green LEDs, the vendor determines which of those indicators to include and selects the method to use for these other indicators to differentiate the module from its steady-state of operation.

This service supersedes use of applicable LEDs for any other service or state in any other CP 2/x, during the time this service is active. When the service is no longer active, the LEDs shall return to normal operation.

6.2.1.3.30 Multiple_Service_Packet

6.2.1.3.30.1 Service overview

The Multiple_Service_Packet service performs a set of services as an autonomous sequence.

6.2.1.3.30.2 Service primitives

The service parameters for this service are shown in Table 50.

Table 50 – Multiple_Service_Packet service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Number of Services	M	M(=)		
Service Offsets	M	M(=)		
Service List	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Number of Responses			M	M(=)
Response Offsets			M	M(=)
Response List			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Number of Services

This parameter indicates the number of embedded services in the Service List parameter below.

Service Offsets

This parameter contains the offsets to the start of each embedded service in the Service List parameter below.

Service List

This parameter contains an array of service request structures.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Number of Responses

This parameter indicates the number of embedded service responses in the Response List parameter below.

Response Offsets

This parameter contains the offsets to the start of each embedded service response in the Response List parameter below.

Service List

This parameter contains an array of service response structures.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.1.3.30.3 Service procedure

The following list details requirements associated with the Multiple_Service_Packet service.

- Shall only be supported by the Message Router object.
- Performs services as an autonomous sequence of services.
- Performs services synchronously in the sequence supplied.
- Performs all services, reporting individual responses for each one. The Message Router object shall indicate to the object the response size available, decreasing the response size after each response, based on the size of the prior response. The target object shall limit its response to the available response size, which may cause an error response such as Reply Data Too Large (0x11) to be returned.

  Four bytes shall be reserved by the Message Router in the response packet for each embedded service to satisfy this requirement (Reply service code byte, pad byte, General Status byte and Additional Status size byte).
- Packs responses into the response buffer in the sequence in which they were executed.
- A response timeout shall be implemented for those service requests that do not guarantee a response.
- Each embedded service shall return a success or failure, as indicated in the response structure. If one or more service requests result in an error, this service shall return an error. The error code returned shall be 0x1E (Embedded service error).

This service allows clients to submit a sequence of “embedded” services in a single message packet. The object processing the Multiple_Service_Packet shall not perform any other service until the entire sequence of embedded services has been attempted.

The embedded services are formatted according to their definitions. Each embedded service shall contain a path.

6.2.2 Connection manager ASE

6.2.2.1 Overview

The Connection Manager ASE (object) is used to manage the establishment and maintenance of communication connections. Every device shall implement instance 1 of the Connection Manager object.

(*) in front of an attribute or a service means that this attribute/service is either mandatory or optional, based on some constraints defined in the attribute/service description.

6.2.2.2 Connection manager class specification

6.2.2.2.1 Connection manager formal model

6.2.2.2.1.1 Class definition

FAL ASE:			**FAL Management ASE**
CLASS:			**Connection_Manager_Object**
CLASS ID:			**6**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
2	(o)	Attribute:	Max Instance
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(o)	Attribute:	OpenReqs
2	(o)	Attribute:	OpenFormat Rejects
3	(o)	Attribute:	OpenResource Rejects
4	(o)	Attribute:	OpenOther Rejects
5	(o)	Attribute:	CloseReqs
6	(o)	Attribute:	CloseFormat Rejects
7	(o)	Attribute:	CloseOther Rejects
8	(o)	Attribute:	ConnTimeouts
9	(o)	Attribute:	Connection Entry List
9.1	(m)	Attribute:	NumConnEntries
9.2	(m)	Attribute:	ConnOpenBits
11	(o)	Attribute:	CpuUtilization
12	(o)	Attribute:	MaxBuffSize
13	(o)	Attribute:	BufSize Remaining
14	(o)	Attribute:	Max Connection Establishment Time
15	(o)	Attribute:	I/O Packets per second
16	(o)	Attribute:	Percent I/O Utilization
17	(o)	Attribute:	Explicit Packets per second
18	(*)	Attribute:	Missed I/O Packets
19	(o)	Attribute:	Type 2 I/O Connections
20	(o)	Attribute:	Type 2 Explicit Connections
SYSTEM MANAGEMENT SERVICES:			
1	(o)	Mgt Service:	Get_Attributes_All
14	(*)	Mgt Service:	Get_Attribute_Single
OBJECT MANAGEMENT SERVICES:			
1	(o)	Ops Service:	Get_Attributes_All
2	(o)	Ops Service:	Set_Attributes_All
14	(*)	Ops Service:	Get_Attribute_Single
16	(o)	Ops Service:	Set_Attribute_Single
OBJECT SPECIFIC SERVICES:			
84	(*)	OpsService:	CM_Open/CM_Forward_Open

78	(*)	OpsService:	CM_Close/CM_Forward_Close
82	(*)	OpsService:	CM_Unconnected_Send
86	(o)	OpsService:	CM_Get_Connection_Data
87	(o)	OpsService:	CM_Search_Connection_Data
90	(*)	OpsService:	CM_Get_Connection_Owner
91	(o)	OpsService:	CM_Large_Forward_Open

6.2.2.2.1.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.2.2.1.3 Object management attributes

OpenReqs

Number of Open requests received including Null Open requests.

This volatile instance attribute has an access rule of Set.

OpenFormat rejects

Number of Open requests rejected by this node due to format errors.

This volatile instance attribute has an access rule of Set.

OpenResource rejects

Number of Open requests rejected by this node.

This volatile instance attribute has an access rule of Set.

OpenOther rejects

Number of Open requests rejected or timed out by downstream nodes.

This volatile instance attribute has an access rule of Set.

CloseReqs

Number of Close requests received.

This volatile instance attribute has an access rule of Set.

CloseFormat rejects

Number of Close requests rejected by this node due to format errors.

This volatile instance attribute has an access rule of Set.

CloseOther rejects

Number of Close requests rejected or timed out by downstream nodes.

This volatile instance attribute has an access rule of Set.

ConnTimeouts

Number of connections which have been timed out by this node after they were opened.

This volatile instance attribute has an access rule of Set.

A device may reject a Set request to any of the eight attributes listed above, using the General Status code "Invalid attribute value", if the attribute value sent is not zero.

Connection entry list

List of connections.

This volatile instance attribute has an access rule of Get only.

NumConnEntries

Number of entries in the ConnOpenBits Attribute.

ConnOpenBits

List of connection data which may be individually queried. Each entry represents a possible connection, and has one of two states: No Connection or Connection Established. More information on a connection can be obtained using a dedicated service.

CpuUtilization

CPU utilization in tenths of a percent.

This attribute indicates the percentage of capacity of the product's compute engine (whether a CPU and/or a core of a CPU) that is most important to the performance of the product. Due to the variations in architectures of various products, each product needs to define this value based on its design, hence the method of calculating this value is vendor specific.

This value is intended to indicate the current loading of the device and may be approaching a point of saturation and performance degradation. This is not simply a measure of how many packets per second are flowing to/from this device, though this is typically a factor in the calculation. It should factor in all CPU processing resources used by the device.

In products that support an operating system, this is often provided by the operating system as the inverse of the time the idle task runs, but that may or may not be a good indication of what's really left for communications related functions. For devices where this is not provided by the operating system already, or there is no operating system used, the vendor shall determine which internal resources should be monitored to determine what can indicate when a device is reaching saturation, and then turn that into a percentage of the max.

EXAMPLE

Here are examples of what the percentages mean:

- 95 % to 100 %: device is overloaded;
- 80 % to 95 %:device is approaching a point of saturation and/or performance degradation;
- below 80 %: device is OK.

Vendors shall put in their documentation factors that will cause the device to go above 80 %.

This volatile instance attribute has an access rule of Get only.

MaxBuffSize

Amount of buffer space originally available (buffer size is in octets).

This volatile instance attribute has an access rule of Get only.

BufSize remaining

Amount of buffer space available at this time (buffer size is in octets).

This volatile instance attribute has an access rule of Get only.

Max Connection Establishment Time

Worst case Forward_Open response time for any connection path of the device. It is recommended that devices whose Max Connection Establishment Time is greater than four seconds support this attribute. See IEC 61158-6-2, 4.1.6.6 for more information.

When this attribute is implemented, a corresponding entry keyword shall be included in the EDS file of the device.

This non-volatile instance attribute has an access rule of Get only.

I/O Packets per second

Indicates the total number of I/O (implicit) packets produced and consumed by this device in the previous second, including duplicate packets.

This volatile instance attribute has an access rule of Get only.

Percent I/O Utilization

Indicates what percentage of the I/O (implicit) communications resources are in use in this device.

NOTE This value is calculated based on the contents of the field "Traffic Specs" for various packet sizes in the EDS file of the device.

This volatile instance attribute has an access rule of Get only.

Explicit Packets per second

Indicates the total number of explicit (unconnected and connected) packet requests and replies sent and received by this device in the previous second, including duplicate packets.

This volatile instance attribute has an access rule of Get only.

Missed I/O Packets

For each connection, as long as the connection remains open, the consumer will subtract the previously received Encapsulation Sequence Number from the current one. If the value exceeds positive one (+1), then that value minus one is the number of packets that have been missed. A connection timeout does not cause Missed I/O Packets to increment.

This volatile instance attribute has an access rule of Set.

A device may reject a Set request to this attribute, using the General Status code "Invalid attribute value", if the attribute value sent is not zero.

This attribute is optional for devices that have an Ethernet-based interface. Devices with no Ethernet-based interface are not permitted to implement this attribute.

Type 2 I/O Connections

Contains the count of open Type 2 I/O (implicit) connections. This value includes connections that have an endpoint in this device, as well as routed connections that go through this device and out any other port.

This volatile instance attribute has an access rule of Get only.

Type 2 Explicit Connections

Contains the count of the open Type 2 Explicit Messaging connections (Class 3 to Message Router). This value includes connections that have an endpoint in this device, as well as routed connections that go through this device and out any other port.

This volatile instance attribute has an access rule of Get only.

6.2.2.2.1.4 System management (class level) and object management (instance level)services

Get_Attribute_Single

This service is **mandatory** if any attributes are implemented and the Get_Attributes_All service is not supported, else it is **optional**.

6.2.2.2.1.5 Object specific services

These Connection Manager services are available only through the Unconnected Message Manager AR (UCMM). They are used either to establish/de-establish an application connection, or to send messages through routers without an established connection.

CM_Open

This service is used by connection originator users to establish an application connection.

CM_Close

This service is used by connection originator users to de-establish an application connection.

CM_Unconnected_Send

The CM_Unconnected_Send service allows an application to send a message to a device through multiple links without first setting up a connection. This optional service is mandatory for originator devices and devices that route messages between links.

CM_Get_Connection_Data

This service is used for diagnostics of a network, to retrieve connection characteristics.

CM_Search_Connection_Data

This service is used for diagnostics of a network, to retrieve connection characteristics.

CM_Get_Connection_Owner

This service is used to determine the owner of a redundant connection. It is mandatory in any device that accepts redundant connection.

6.2.2.3 Connection manager ASE service specification

6.2.2.3.1 Supported services

Subclause 6.2.2.3 contains the definition of services that are unique to this ASE. The services defined for this ASE are

CM_Open
CM_Close
CM_Unconnected_Send
CM_Get_Connection_Data
CM_Search_Connection_Data

CM_Get_Connection_Owner

6.2.2.3.2 Connection manager common service parameters

The following service parameters are common to all Connection Manager services.

The parse order of the parameters is device specific, although checking the Electronic Key segment if present in the connection path first is highly recommended..

NOTE 1 Complete specification of these parameters is provided in IEC 61158-6-2.

Originator local ID

This parameter is locally generated by the originator node, and identifies locally this invocation of the service. It is used to associate service confirmations with requests. Therefore, no two outstanding service invocations may be identified by the same ID value.

Target local ID

This parameter is locally generated by the target node, and identifies locally this invocation of the service. It is used to associate service responses with indications. Therefore, no two outstanding service invocations may be identified by the same ID value.

Path

This parameter specifies the FAL APO or FAL APO element that is the target of a connection establishment (or de-establishment) request, or an unconnected message request. The path shall contain multiple segments which can indicate the network route to the node containing the FAL APO (in the case of multiple links), the identification of the APO element within the target node (Routing Path), and optional additional information for the target APO (Additional Path). This Additional Path parameter is passed on to the target application, and could be a data segment used to configure the application, the transport class, and the trigger.

Transport identifier

The Transport identifier parameter is generated locally by the originator Connection Manager, and shall serve as a further reference for the user to the newly established connection.

Transaction_priority

The Transaction_priority parameter determines which data-link layer priority to use for the messages that convey the service requests and responses, and shall be one of LOW or HIGH.

NOTE 2 The definition of DLL priority is specified in IEC 61158-4-2.

Transaction_timeout

The Transaction_timeout parameter determines the time to wait for service completion. The units for Transaction_timeout is milliseconds.

O2T_ConnParm / T2O_ConnParm

The O2T parameter specifies the characteristics of the originator to target (O⇒T) communication on the new connection. The T2O parameter specifies the characteristics of the target to originator (T⇒O) communication.

CM_RPI

The Requested Packet Interval (CM_RPI) specifies the expected time between packets, i.e. how frequently the originating application requires the transmission of data from the target application. It shall be expressed as a number of microseconds.

Type

The Type parameter indicates the type of the connection, and shall be one of NULL, MULTIPOINT, or POINT2POINT.

A value of NULL indicates that the specified direction (O⇒T or T⇒O) has no transport associated (see IEC 61158-6 for use of the NULL connection type).

A value of MULTIPOINT indicates that other connections may later participate in the transport. Data is sent using a destination address that can be received by multiple consumers. The same underlying transport shall be reused by subsequent MULTIPOINT connections that specify the same application path.

A value of POINT2POINT indicates that this connection shall not allow any other transport to participate (data is sent only between one producer and one consumer).

Priority

The Priority parameter determines which data-link layer priority to use for the new transports, and shall be one of LOW, HIGH, SCHEDULED or URGENT.

Variable

The Variable parameter specifies whether every application data which traverses the new transport shall be the same size or not, and shall be either TRUE (variable size) or FALSE (fixed size).

Size

The Size parameter specifies the size (in octets) of the largest application data packet for this connection.

Redundant owner

The Redundant Owner parameter specifies whether more than one owner may be permitted to make a connection simultaneously.

CM_RPI_multiplier

The product of the CM_RPI_multiplier parameter and the requested packet interval (CM_RPI) specifies the time-out for the transport. If the application has not used the transport within this time-out, the transport shall close, which shall subsequently close the connection.

Trigger

The trigger parameter specifies the trigger mode, and shall be one of CYCLIC, CHANGE_OF_STATE, or APPLICATION.

NOTE 3 This parameter is used to configure the transport(s). Its meaning to the transports is described in 6.3.1.

Trans_class

The Trans_class parameter specifies the transport class selected, and shall be one of NULL, DUPLICATE_DETECTION, ACKNOWLEDGED, VERIFIED, NON-BLOCKING, NON-BLOCKING_FRAGMENTING, or MULTIPOINT_FRAGMENTING

NOTE 4 This parameter is used to configure the transport(s). Its meaning to the transports is described in 6.3.1.

Is_server

The Is_server parameter specifies if the new transport shall be of server (TRUE) or a client (FALSE).

NOTE 5 This parameter is used to configure the transport(s). Its meaning to the transports is described in 6.3.1.

Originator_Vendor_ID

Vendor ID of the device which has requested establishment of the connection (connection originator). This is a reference to the corresponding attribute in instance #1 of its Identity Object.

Originator_Serial_Number

Serial number of the device which has requested establishment of the connection (connection originator). This is a reference to the corresponding attribute in instance #1 of its Identity Object.

Connection serial number

The Connection Serial Number parameter is a value selected by the originator Connection Manager to uniquely identify a connection within the originator device.

Connection triad

The Connection triad relates to the combination of Connection Serial Number, Originator_Vendor_ID and Originator_Serial_Number parameters.

Service status

This parameter provides information on the result of service execution. It is returned in all Connection Manager service response/confirmation primitives. It has the same definition as the Service Status parameter of the FAL Management services. Corresponding available Status Codes and Extended Status formats are detailed in 6.2.2.3.3.

O2T_CM_API / T2O_CM_API

The O2T_CM_API and T2O_CM_API specify the actual packet interval that shall be used for the new connection, i.e. how frequently the connection produces its data. It shall be expressed in microseconds. These values may be different than the requested packet intervals, but shall always be equal or smaller than the requested interval.

Remaining path size

In the failure response, the remaining_path parameter specifies the “pre-stripped” size. This is the size of the path when the node first receives the request and has not yet started processing it. A target node may instead return 0 (zero) for this parameter. Associated with the Service Status (Status Code and Extended Status), this parameter allows to identify the type and location of any errors found during connection establishment or de-establishment.

Response data

If the target has any application specific response data, it shall be returned in the Response Data parameter.

6.2.2.3.3 Connection manager service status codes

Specific status codes for the Connection Manager are detailed in IEC 61158-6-2.

6.2.2.3.4 CM_Open / CM_Forward_Open / CM_Large_Forward_Open

6.2.2.3.4.1 Service overview

The CM_Open / CM_Forward_Open service is used by connection originator users to establish an application connection to a specified target object instance or instance element. The CM_Open request contains the parameters to select connection types, transports and to specify the requested packet interval in both the originator to target and target to originator direction.

The CM_Large_Forward_Open has the same function and same behaviour as the CM_Open / CM_Forward_Open except that it allows the establishment of connections larger than 511 octets.

6.2.2.3.4.2 Service primitives

The service parameters for this service are shown in Table 51.

Table 51 – CM_Open service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Additional Path	U	U(=)		
Transport identifier	M			
Transaction_priority	M			
Transaction_timeout	M			
O2T_ConnParm	M	M(=)		
CM_RPI	M	M(=)		
Type	M	M(=)		
Priority	M	M(=)		
Variable	M	M(=)		
Size	M	M(=)		
T2O_ConnParm	M	M(=)		
CM_RPI	M	M(=)		
Type	M	M(=)		
Priority	M	M(=)		
Variable	M	M(=)		
Size	M	M(=)		
CM_RPI_multiplier	M	M(=)		
Trigger	M	M(=)		
Trans_class	M	M(=)		
Is_server	M	M(=)		
Originator_Vendor_ID		M		
Originator_Serial_Number		M		
Connection Serial Number		M		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Transport identifier				M
O2T_CM_API			M	M(=)
T2O_CM_API			M	M(=)
Remaining Path Size			M	M(=)
Response Data			U	U(=)

Argument

The argument contains the parameters of the service request.

Transport identifier

The transport identifier parameter in the request allows a previously opened connection to be reused (this is the same identifier as the one returned in the CM_Open confirmation for this first connection). If Transport identifier is zero, then a new transport shall be created.

NOTE Reusing a transport allows an originator to establish multipoint connections in the O⇒T direction.

Result

This parameter indicates whether the service request succeeded or failed.

6.2.2.3.4.3 Service procedure

The CM_Open request primitive, sent from the originating application, first triggers the connection establishment process into the local Connection Manager, before being forwarded to the remote target Connection Manager (using a CM_Forward_Open service request).

The corresponding CM_open indication primitive shall then be sent to the target application from the target Connection Manager, after some local processing. The application shall reply to the CM_open indication primitive with a CM_open response whether the application accepts the connection or not.

After the connection has been established, or if the connection attempt failed, a CM_Forward_Open service response is sent from the target Connection Manager to the originator Connection Manager, then a CM_open confirm primitive shall be sent from the local originator Connection Manager back to the originating application. The confirmation shall indicate the status of the connection and shall provide information about the connection including the Transport Identifier parameter, which shall serve as a reference to the newly established connection.

6.2.2.3.5 CM_Close / CM_Forward_Close

6.2.2.3.5.1 Service overview

The CM_Close service is used by connection originator users to de-establish ("close") an application connection previously established to a target object instance or instance element, and to de-allocate the resources associated with the connection. The service request shall indicate the connection to close by using the original path and the Transport identifier.The originating application shall save both the connection path required to establish the connection, and the Transport identifier returned by the CM_open confirmation.

6.2.2.3.5.2 Service primitives

The service parameters for this service are shown in Table 52.

Table 52 – CM_Close service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Additional Path	U	U(=)		
Transport identifier	M			
Transaction_priority	M			
Transaction_timeout	M			
Originator_Vendor_ID		M		
Originator_Serial_Number		M		
Connection Serial Number		M		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Transport identifier				M
Remaining Path Size			M	M(=)
Response Data			U	U(=)

Argument

The argument contains the parameters of the service request.

Result

This parameter indicates whether the service request succeeded or failed.

6.2.2.3.5.3 Service procedure

The CM_close request primitive, sent from the originator application, first triggers the connection de-establishment process into the local Connection Manager, before being forwarded to the remote target Connection Manager (using a CM_Forward_Close service request). of a connection to de-establish the connection and de-allocate the resources associated with the connection. The service shall be sent from the originator application to the local Connection Manager.

The corresponding CM_close indication primitive shall then be sent from the target Connection Manager to the target application, after some local processing.

A successful CM_close request and response shall result in all non-shared resources associated with this connection in all nodes participating in the original connection being released, including connection IDs, bandwidth allocation, and internal memory buffers. Shared resources are those still in use by other connection(s) due to multicast. A router should also release non-shared resources if the response received indicates an error of General Status 0x01 and Extended Status 0x0107 (Target Connection Not Found).

For a CM_close service request received by a target node to be successful, it is sufficient that the Connection Triad matches an existing connection's parameters. If there is no connection match, no connection is released and the target shall return an error with a General Status code of 0x01 and an Extended Status code of 0x0107 (Target Connection Not Found), unless

the device detected an improper Connection Path. The Connection Path Size parameters may be ignored by the target node. However, an improperly formatted Connection_Path or Connection_Path mismatch may cause the service to be unsuccessful and return an error regardless of a Connection Triad match (see connection path error conditions below).

A CM_close service request received by an intermediate node along the path between the originator and target nodes shall be forwarded regardless of a Connection Triad match (in the case of a match not found, the connection may have timed out at this intermediate node). An improperly formatted Connection_Path shall, and a Connection_Path mismatch, may cause the service to be unsuccessful and return an error (see connection path error conditions below).

If either a target or intermediate node does detect a connection path error condition, it shall return an error response as described below. In all cases the connection remains unchanged (the connection is not released).

Improperly formatted Connection_Path:

- If a node detects that the value of the Connection_Path_Size field is smaller than the size of the Connection_Path, a General Status code of 0x15 shall be returned indicating that too much data is present in the service request.
- If a node detects that the value of the Connection_Path_Size field is greater than the size of the Connection_Path, General Status code 0x13 shall be returned indicating that not enough data is present in the service.

Connection_Path mismatch:

- If the node detects a mismatch in the Connection_Path between the value received in the CM_close request and the value sent in the originating connection request, a General Status code of 0x01 and an Extended Status code of either 0x316 (Error In Forward Close Service Connection Path Mismatch) or 0x0315 (Invalid Segment in Connection Path) shall be returned.

Success shall be returned when the connection has been deleted at the target. The originator, and each intermediate node along the path, closes the connection and releases resources associated with that connection when the success response is received.

6.2.2.3.6 CM_Unconnected_Send

6.2.2.3.6.1 Service overview

The CM_Unconnected_Send service allows an application to send a message to a device through multiple links without first setting up a connection. This optional service is mandatory for originator devices and devices that route messages between links.

6.2.2.3.6.2 Service primitives

The service parameters for this service are shown in Table 53.

Table 53 – CM_ Unconnected_Send service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M		
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Additional Path	U	U(=)		
Transaction_priority	M			
Transaction_timeout	M			
OM_Service Code	M	M(=)		
OM_Service Request Parameters	M	M(=)		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Remaining Path Size			M	M(=)
OM_Service Response Parameters			M	M(=)

Argument

The argument contains the parameters of the service request.

OM_Service Code

This parameter indicate the code of the Object Management service to be performed by the target element.

OM_Service request parameters

This parameter contains the specific arguments parameters of the Object Management service to be performed by the target element.

Result

This parameter indicates whether the service request succeeded or failed.

OM_Service response parameters

This parameter contains the specific result parameters of the Object Management service to be performed by the target element.

6.2.2.3.6.3 Service procedure

The CM_Unconnected_Send service shall use the Connection Manager object in each intermediate node to forward the message and to remember the return path. The UCMM of each link shall be used to forward the request from Connection Manager to Connection Manager just as it is for the Forward_Open service; however. no connection shall be built. The CM_Unconnected_Send service shall be sent to the local Connection Manager and shall be sent between intermediate nodes. When an intermediate node removes the last port segment, the message shall be formatted as a UCMM message and sent to the port and link address of the last segment.

NOTE The target node never sees the CM_Unconnected_Send service but only a standard message arriving via the UCMM.

The CM_Unconnected_Send response shall be generated by the last intermediate node from the UCMM response generated by the target node or by an intermediate node as the result of a UCMM time-out, a problem with the embedded message, or a problem with the Unconnected Service Request itself. The packet shall be routed from intermediate node to intermediate node using the information stored when the CM_Unconnected_Send request was processed. The response shall contain status information about the request and a response generated by the target node.

6.2.2.3.7 CM_Get_Connection_Data

6.2.2.3.7.1 Service overview

The CM_Get_Connection_Data service is used for diagnostics of a network. This service shall return the parameters associated with a specified connection.

6.2.2.3.7.2 Service primitives

The service parameters for this service are shown in Table 54.

Table 54 – CM_Get_Connection_Data service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Target transport identifier	M	M(=)		
Transaction_priority	M			
Transaction_timeout	M			
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Target transport identifier			M	M(=)
Connection state			M	M(=)
Originator port			M	M(=)
Target port			M	M(=)
Connection Serial Number			M	M(=)
Originator_Vendor_ID			M	M(=)
Originator_Serial_Number			M	M(=)
Originator O2T_CID			M	M(=)
Target O2T_CID			M	M(=)
O2T_CM_RPI_multiplier			M	M(=)
Originator O2T_CM_RPI			M	M(=)
Originator O2T_CM_API			M	M(=)
Originator T2O_CID			M	M(=)
Target T2O_CID			M	M(=)
T2O_CM_RPI_multiplier			M	M(=)
Originator T2O_CM_RPI			M	M(=)
Originator T2O_CM_API			M	M(=)

Argument

The argument contains the parameters of the service request.

Target transport identifier

This parameter identifies the connection for which data is requested. This number may be different from device to device even for the same connection. This number corresponds to the offset into the Connection Manager attribute that enumerates the status of the connections.

Result

This parameter indicates whether the service request succeeded or failed.

Connection serial number

This parameter contains the serial number of the established connection.

Originator_Vendor_ID

Originator_Serial_Number

These parameters identify the originating node for the connection.

6.2.2.3.7.3 Service procedure

No specific service procedure.

6.2.2.3.8 CM_Search_Connection_Data

6.2.2.3.8.1 Service overview

The CM_Search_Connection_Data service is used for diagnostics of a network. This service shall return the parameters associated with a specified connection within a device, identified by vendor and serial number.

6.2.2.3.8.2 Service primitives

The service parameters for this service are shown in Table 55.

Table 55 – CM_Search_Connection_Data service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Transaction_priority	M			
Transaction_timeout	M			
Originator_Vendor_ID	M	M(=)		
Originator_Serial_Number	M	M(=)		
Connection Serial Number	M	M(=)		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Target transport identifier			M	M(=)
Connection state			M	M(=)
Originator port			M	M(=)
Target port			M	M(=)
Connection Serial Number			M	M(=)
Originator_Vendor_ID			M	M(=)
Originator_Serial_Number			M	M(=)
Originator O2T_CID			M	M(=)
Target O2T_CID			M	M(=)
O2T_CM_RPI_multiplier			M	M(=)
Originator O2T_CM_RPI			M	M(=)
Originator O2T_CM_API			M	M(=)
Originator T2O_CID			M	M(=)
Target T2O_CID			M	M(=)
T2O_CM_RPI_multiplier			M	M(=)
Originator T2O_CM_RPI			M	M(=)
Originator T2O_CM_API			M	M(=)

Argument

The argument contains the parameters of the service request.

Connection serial number

This parameter contains the serial number of the established connection.

Originator_Vendor_ID

Originator_Serial_Number

These parameters identify the originating node for the connection.

Result

This parameter indicates whether the service request succeeded or failed. Its contents are identical to the Get_Connection_Data primitives.

6.2.2.3.8.3 Service procedure

No specific service procedure.

6.2.2.3.9 CM_Get_Connection_Owner

6.2.2.3.9.1 Service overview

The CM_Get_Connection_Owner service returns data about the connection(s) that own(s) a particular object. It shall be implemented in any device that accepts redundant connections.

6.2.2.3.9.2 Service primitives

The service parameters for this service are shown in Table 56.

Table 56 – CM_Get_Connection_Data service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Transaction_priority	M			
Transaction_timeout	M			
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Number of connections			M	M(=)
Number claiming ownership			M	M(=)
Number ready for ownership			M	M(=)
Last action			M	M(=)
Connection Serial Number			M	M(=)
Originator_Vendor_ID			M	M(=)
Originator_Serial_Number			M	M(=)

Argument

The argument contains the parameters of the service request.

Path

This parameter specifies the internal path from the Message Router in the target node to the selected object. It shall be the same path as would have appeared in a CM_Forward_Open request for this object, except that corresponding Additional Path electronic key, network, and data segments shall be removed before matching the paths.

Result

This parameter indicates whether the service request succeeded or failed.

Number of connections

This parameter contains the number of connections currently open to the specified path.

Number claiming ownership

This parameter contains the number of connections that are currently claiming ownership.

Number ready for ownership

This parameter contains the number of connections currently ready for ownership.

Last action

This parameter indicates mode of owning connection as follows. If the owning connection is in the idle mode, the Last action field shall equal 1. If the connection is in the run mode, the Last action field shall equal 2. If there is currently no owner, the Last action field shall equal 0. Values of the Last action field from 0x03 through 0xFF are reserved. If an implementation does not support one of these fields, the field shall equal 0xFF.

Connection serial number

Originator_Vendor_ID

Originator_Serial_Number

These parameters shall be from the CM_Forward_Open request whose connection is currently the owner of the object. If no owner currently exists, these parameters shall each be set to zero.

6.2.2.3.9.3 Service procedure

No specific service procedure.

6.2.3 Connection ASE

6.2.3.1 Overview

The Connection ASE (object) allocates and manages the internal resources associated with both I/O and Explicit Messaging Connections. The specific instance generated by the Connection ASE is referred to as a Connection instance or a Connection object.

A Connection object within a particular module actually represents one of the end-points of a connection. It is possible for one of the connection end-points to be configured and “active” (e.g. transmitting) without the other end-point(s) being present. Connection objects are used to model the communication specific characteristics of a particular application-to-application(s) relationship.

A specific Connection object Instance manages the communication specific aspects related to an end-point. A Connection object uses the services provided by a Link Producer and/or Link Consumer to perform low-level data transmission and reception functions.

6.2.3.2 Connection class specification

6.2.3.2.1 Connection formal model

6.2.3.2.1.1 Class definition

(*) in front of an attribute or a service means that this attribute/service is either mandatory or optional, based on some constraints defined in the attribute/service description.

FAL ASE:			**Connection ASE**
CLASS:			**Connection _Object**
CLASS ID:			**5**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
2	(o)	Attribute:	Max Instance
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
8	(*)	Attribute	Connection Request Error Count
9	(*)	Attribute	Safety Connection Counters
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(m)	Attribute:	State
2	(m)	Attribute:	Instance_type
3	(m)	Attribute:	TransportClass_trigger
4	(o)	Attribute:	CP2/3_produced_connection_id
5	(o)	Attribute:	CP2/3_consumed_connection_id
6	(o)	Attribute:	CP2/3_initial_comm_characteristics
7	(m)	Attribute:	Produced_connection_size
8	(m)	Attribute:	Consumed_connection_size
9	(m)	Attribute:	Expected_packet_rate
10	(o)	Attribute:	CPF2_produced_connection_id
11	(o)	Attribute:	CPF2_consumed_connection_id
12	(o)	Attribute:	Watchdog_timeout_action
13	(o)	Attribute:	Produced_connection_path_ length
14	(o)	Attribute:	Produced_connection_path
15	(o)	Attribute:	Consumed_connection_path_ length
16	(o)	Attribute:	Consumed_connection_path
17	(o)	Attribute:	Production_inhibit_time
18	(o)	Attribute:	Connection_timeout_multiplier
19	(o)	Attribute;	Connection_binding_list
SYSTEM MANAGEMENT SERVICES:			
5	(o)	Mgt Service:	Reset
8	(o)	Mgt Service:	Create
9	(o)	Mgt Service:	Delete
14	(*)	Mgt Service:	Get_Attribute_Single
17	(o)	Mgt Service:	Find_Next_Object_Instance
OBJECT MANAGEMENT SERVICES:			
5	(o)	Ops Service:	Reset
9	(o)	Ops Service:	Delete
13	(o)	Ops Service:	Apply_Attributes
14	(m)	Ops Service:	Get_Attribute_Single
16	(o)	Ops Service:	Set_Attribute_Single
OBJECT SPECIFIC SERVICES:			
75	(*)	Mgt Service:	Connection_Bind
76	(*)	Mgt Service:	Producing_Application_Lookup
84	(*)	Mgt Service:	SafetyOpen
78	(*)	Mgt Service:	SafetyClose

6.2.3.2.1.2 System management attributes (class attributes)

Connection request error count

This attribute is used for safety (see IEC 61784-3-2).

Safety connection counters

This attribute is used for safety (see IEC 61784-3-2).

6.2.3.2.1.3 Object management attributes

Access rules for all Connection object management attributes are specified in 6.2.3.2.1.4.

State

State of the object.

Instance_type

Indicates either I/O or Messaging Connection.

TransportClass_trigger

Defines behavior of the Connection.

CP2/3_produced_connection_id

Placed in CAN Identifier field when the Connection transmits on a CP 2/3 subnet.

CP2/3_consumed_connection_id

CAN Identifier field value that denotes message to be received on a CP 2/3 subnet.

CP2/3_initial_comm_characteristics

Defines the Message Group(s) across which productions and consumptions associated with this Connection occur on a CP 2/3 subnet.

Produced_connection_size

Maximum number of octets transmitted across this Connection.

Consumed_connection_size

Maximum number of octets received across this Connection.

Expected_packet_rate

Defines timing associated with this Connection.

CPF2_produced_connection_id

Identifies the message sent on the subnet by this connection.

CPF2_consumed_connection_id

Identifies the message received from the subnet for this connection.

Watchdog_timeout_action

Defines how to handle Inactivity/Watchdog timeouts.

Produced_connection_path_ length

Number of octets in the produced_connection_path attribute.

Produced_connection_path

Specifies the application object(s) whose data is to be produced by this Connection object.

Consumed_connection_path_ length

Number of octets in the consumed_connection_path attribute.

Consumed_connection_path

Specifies the application object(s) that are to receive the data consumed by this Connection object.

Production_inhibit_time

Defines minimum time between new data production. This attribute is required for all I/O Client connections, except those with a production trigger of Cyclic.

Connection_timeout_multiplier

Specifies the multiplier applied to the expected_packet_rate value to derive the value for the Inactivity/Watchdog Timer.

Connection_binding_list

List of I/O connection instances bound to this instance.

6.2.3.2.1.4 System management (class level) and object management (instance level) services

Get_Attribute_Single

At the class level, this service is **mandatory** if any attributes are implemented, else it is **optional**.

Connection object attribute access rules

During the configuration of a Connection instance using the Set_Attribute service, a module shall perform value checks of each separate attribute when it is modified. If an error is detected, then an Error Response is returned. The discovery of an error **DOES NOT** cause the deletion of the Connection instance. Table 57, Table 58 and Table 59 summarize the access rules for the different connection types

Important: If the produced_connection_id and/or consumed_connection_id attributes contain a non-default value upon reception of the Apply Request, then the related portion of the initial_comm_characteristics attribute is ignored and the ID value is validated and used.

The following series of tables indicates when an attribute can be read or written via a Get and/or Set operation based on the Connection's **state** and **instance_type**.

Important: If a request to Get/Set a supported attribute is received but the current state and/or instance_type of Connection dictates that the requested access is invalid, then the returned error status indicates: *The object cannot perform the requested service in its current mode/state*.

Table 57 – I/O Connection object attribute access

Attribute	I/O connection state				
	Non-existent	**Configuring**	**Waiting for connection ID**	**Established**	**Timed Out**
State	Not available	Get Only	Get Only	Get Only	Get Only
instance_type	Not available	Get Only	Get Only	Get Only	Get Only
transportClass_trigger	Not available	Get/Set	Get Only	Get Only	Get Only
produced_connection_id	Not available	Get/Set	Get/Set	Get/Set	Get/Set
consumed_connection_id	Not available	Get/Set	Get/Set	Get/Set	Get/Set
initial_comm_characteristics	Not available	Get/Set	Get Only	Get Only	Get Only
produced_connection_size	Not available	Get/Set	Get Only	Get Only	Get Only
consumed_connection_size	Not available	Get/Set	Get Only	Get Only	Get Only
expected_packet_rate[a]	Not available	Get/Set	Get Only	Get/Set	Get/Set
watchdog_timeout_action	Not available	Get/Set	Get Only	Get/Set	Get/Set
produced_connection_ path_length	Not available	Get Only	Get Only	Get Only	Get Only
produced_connection_path	Not available	Get/Set	Get Only	Get Only	Get Only
consumed_connection_ path_length	Not available	Get Only	Get Only	Get Only	Get Only
consumed_connection_path	Not available	Get/Set	Get Only	Get Only	Get Only
production_inhibit_time	Not available	Get/Set	Get Only	Get/Set	Get/Set
connection_timeout_multiplier	Not available	Get/Set	Get Only	Get/Set [a]	Get/Set
Connection_binding_list	Not available	Get Only	Get Only	Get Only	Get Only

[a] When a Connection object is in the **Established** state, any modifications to the **expected_packet_rate** or **connection_timeout_multiplier** attributes have immediate effect on the Inactivity/Watchdog Timer. The following steps are performed by a Connection object in the **Established** state when a request is received to modify the **expected_packet_rate or connection_timeout_multiplier** attribute:

– the current Inactivity/Watchdog Timer is canceled,

– a new Inactivity/Watchdog Timer is activated based on the new value in the **expected_packet_rate** or **connection_timeout_multiplier** attribute.

Table 58 – Bridged Connection object attribute access

Attribute	Default value [a]	Bridged connection state			
		Non-existent	Configuring	Established	Closing
state	1	Not Available	Get Only	Get Only	Get Only
instance_type	2	Not Available	Get Only	Get Only	Get Only
transportClass_trigger	From service	Not Available	Get Only	Get Only	Get Only
produced_connection_id	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_id	From service	Not Available	Get Only	Get Only	Get Only
initial_comm_characteristics	From service	Not Available	Get Only	Get Only	Get Only
produced_connection_size	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_size	From service	Not Available	Get Only	Get Only	Get Only
expected_packet_rate	From service	Not Available	Get Only	Get Only	Get Only
watchdog_timeout_action	1	Not Available	Get Only	Get Only	Get Only
produced_connection_path_length	From service	Not Available	Get Only	Get Only	Get Only
produced_connection_path	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_path_length	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_path	From service	Not Available	Get Only	Get Only	Get Only
production_inhibit_time	From service	Not Available	Get Only	Get Only	Get Only
Connection_timeout_multiplier	From service	Not Available	Get Only	Get Only	Get Only
Connection_binding_list	Empty	Not Available	Get Only	Get Only	Get Only

[a] The default value is either the value indicated or is set based on one of the parameters of the Forward Open service received by the Connection Manager object.

Table 59 – Explicit messaging object attribute access

Attribute	Explicit messaging connection state	
	Non-existent	Established/deferred delete
State	Not available	Get Only
instance_type	Not available	Get Only
transportClass_trigger	Not available	Get Only
produced_connection_id	Not available	Get Only
consumed_connection_id	Not available	Get Only
initial_comm_characteristics	Not available	Get Only
produced_connection_size	Not available	Get Only
consumed_connection_size	Not available	Get Only
expected_packet_rate[1]	Not available	Get/Set [a]
watchdog_timeout_action	Not available	Get/Set
produced_connection_ path_length	Not available	Get Only
produced_connection_path	Not available	Get Only
consumed_connection_ path_length	Not available	Get Only
consumed_connection_path	Not available	Get Only
production_inhibit_time	Not available	Get Only
connection_timeout_multiplier	Not available	Get/Set [a]
Connection_binding_list	Not available	Get Only

[a] When a Connection object is in the Established state, any modifications to the expected_packet_rate or connection_timeout_multiplier attributes have immediate effect on the Inactivity/Watchdog Timer.

The following steps are performed by a Connection object in the Established state when a request is received to modify the expected_packet_rate or connection_timeout_multiplier attribute:

– the current Inactivity/Watchdog Timer is cancelled,

– a new Inactivity/Watchdog Timer is activated based on the new value in the expected_packet_rate or connection_timeout_multiplier attribute.

6.2.3.2.1.5 Object specific services

Connection_Bind

The Connection_Bind object specific service binds two dynamically created I/O connection instances together for purposes of connection timeouts and deletions. This service is **mandatory** if the Create service is supported, else it is **optional**.

Producing_Application_Lookup

The Producing_Application_Lookup object specific service provides a mechanism to find one or more connection instances in the Established state producing data from a given application object. This service is **mandatory** if the Create service is supported, else it is **optional**.

SafetyOpen

See IEC 61784-3-2 for the definition of this service.

SafetyClose

See IEC 61784-3-2 for the definition of this service.

6.2.3.2.1.6 Connection object state machines

See IEC 61158-6-2, Clause 7 (AP-Context state machine).

6.2.3.2.1.7 Specific requirements of the Connection object

Connection Timing

Three *types* of timers are involved in a connection:

- Transmission Trigger Timer
- Inactivity/Watchdog Timer
- Production Inhibit Timer

The first two timers are initialized based on the value in the **expected_packet_rate** attribute.

Important: For Explicit Messaging, the Application is responsible for providing *response timeout* facilities. The amount of time a Client waits for a Server to respond to a request depends on the application and, possibly, on the service. Due to Media Access mechanisms used for accessing the subnet, an implementation should wait until an Explicit Request Message is transmitted before activating any response related timers.

Transmission trigger timer

The Transmission Trigger Timer is required to be implemented by the client endpoint of an I/O Connection. It is optional for the client endpoint of an explicit messaging connection. It shall not be implemented in the server endpoint of any connection.

The Transmission Trigger Timer value shall be equal to the **expected_packet_rate**. Expiration of this timer shall result in the production of the associated data regardless of trigger type or connection class.

The Transmission Trigger Timer/expected_packet_rate establishes the production rate of cyclic connections, and the heartbeat/keep-alive production rate for Change-of-State and application triggered connections.

The Transmission Trigger Timer is restarted with the **expected_packet_rate** value immediately upon the Connection Object being told to produce new data by the application or the Transmission Trigger Timer expiring (see Figure 17).

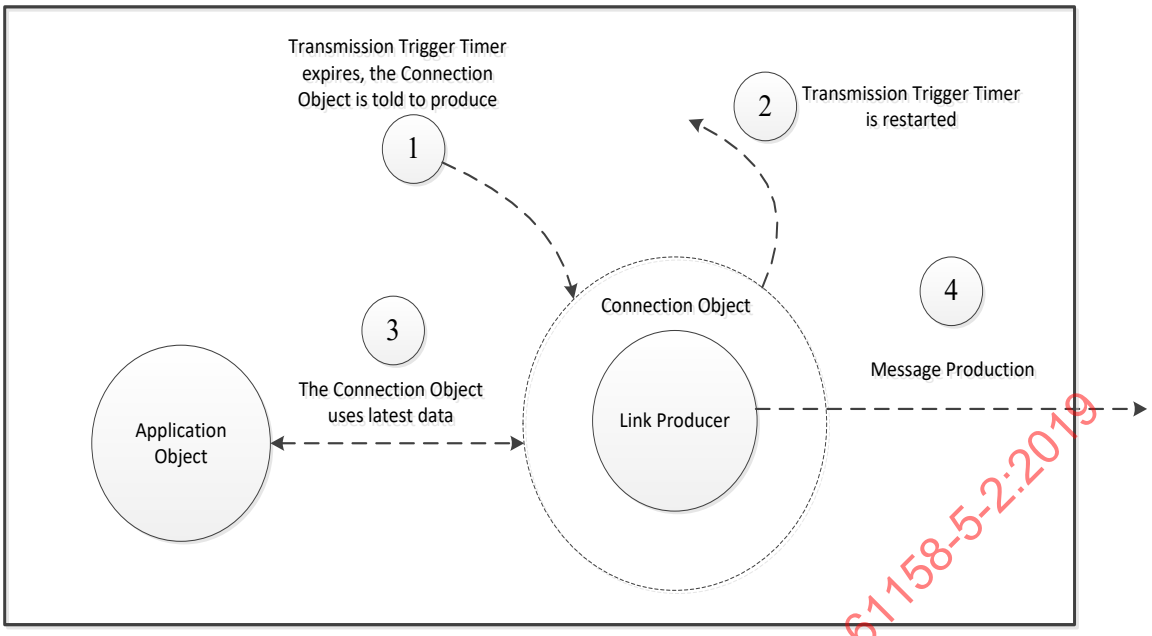


Figure 17 – Transmission Trigger Timer behavior

The Transmission Trigger Timer is activated when the Connection transitions to the **Established** state.

Important: If the expected_packet_rate attribute contains the value zero (0), then the Transmission Trigger Timer is not activated and/or used by the Client end-point.

Inactivity/watchdog timer

This timer is required to be managed by any **consuming** Connection object. Consuming Connection objects include:

- Client end-point Connection objects whose transportClass_trigger attribute indicates either Transport Class 2 or 3.
- All Server end-point Connection objects.

This timer is activated when the Connection transitions to the **Established** state. The tasks listed below are performed by a Connection immediately upon detecting that a valid message has been consumed:

- The current value for the Inactivity/Watchdog Timer is restored to its initial value and the timer is stopped.
- A new Inactivity/Watchdog Timer is activated.

The bullet items above indicate that the new Inactivity/Watchdog Timer is activated before the received message is processed.

Expiration of this timer is an indication that the Connection object has timed out while waiting to consume (see Figure 18). The Connection object performs the following steps when the Inactivity/Watchdog timer expires:

- issues an indication of this event to the application,
- performs the action indicated by the watchdog_timeout_action attribute.

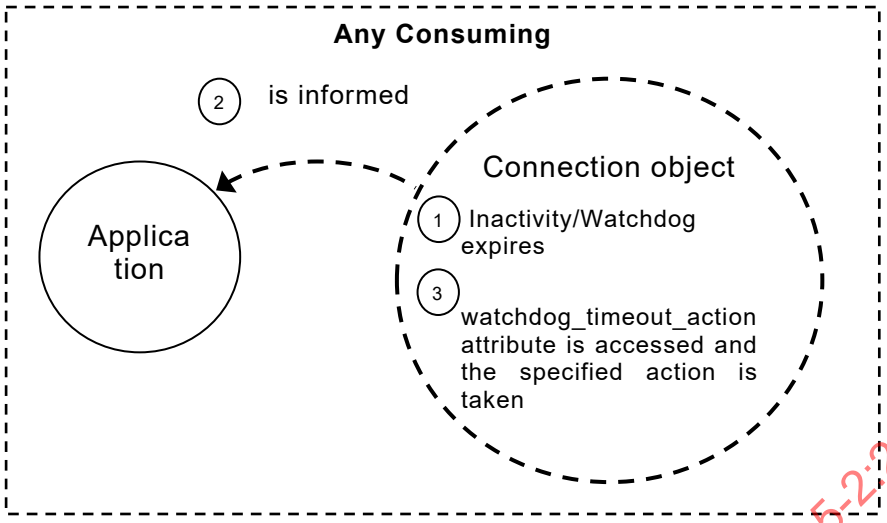


Figure 18 – Inactivity watchdog timer

Two different values are used for the Inactivity/Watchdog Timer, based on whether or not the Connection object has consumed a message:

1) The initial value loaded into the Inactivity/Watchdog Timer is either 10 000 ms (10 s) or the **expected_packet_rate** multiplied by the **connection_timeout_multiplier**, depending on which value is numerically greater.

   NOTE 1 If the **expected_packet_rate** attribute specifies the value zero (0), then the Inactivity/Watchdog Timer is not activated and/or used by the Connection object. A Connection object whose **expected_packet_rate** attribute is zero (0) will never experience an Inactivity/Watchdog timeout.

   If the **expected_packet_rate** attribute multiplied by the **connection_timeout_multiplier** is greater than 10 000, then the **expected_packet_rate** multiplied by the **connection_timeout_multiplier** is used. Otherwise, 10 000 (10 s) is used. This is referred to as the **pre-consumption** timeout value. This value is used because a Connection may transition to the **Established** state before all end-points are fully configured. An example is shown in Figure 19.

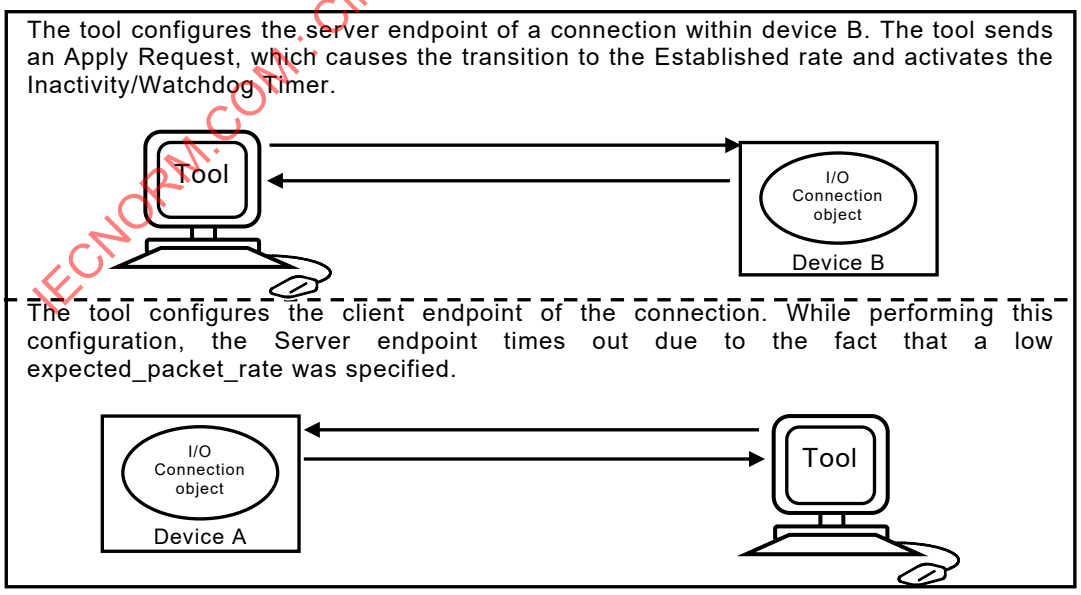


Figure 19 – Using tools for configuration

This rule takes into consideration that the application-to-application connection is not really established until the associated information exchange is performed for the first time.

2) All subsequent activations of the Inactivity/Watchdog Timer use the **expected_packet_rate** multiplied by the **connection_timeout_multiplier** as the number of milliseconds to load into the Inactivity/Watchdog Timer.

NOTE 2 If the **expected_packet_rate** attribute specifies the value zero (0), then the Inactivity/Watchdog Timer is not activated and/or used by the Connection object. A Connection object whose **expected_packet_rate** attribute is zero (0) will never experience an Inactivity/Watchdog timeout.

Production inhibit timer

This timer is required to be managed by the Connection Object within the Client end-point of an I/O Connection when the **production_inhibit_time** attribute value is non-zero. This timer is started only when new data is produced by the Connection Object. Data produced due to the expiration of the Transmission Trigger Timer shall not result in the Production Inhibit Timer being restarted. The Connection object shall not produce new data if this timer is running. An Acknowledge Handler object (see 6.2.1.2.5.7) retry may however be sent to the Link Producer. If one or more new data events occur while this timer is running, the Connection object shall send the most recent new data when this timer expires (see Figure 20).

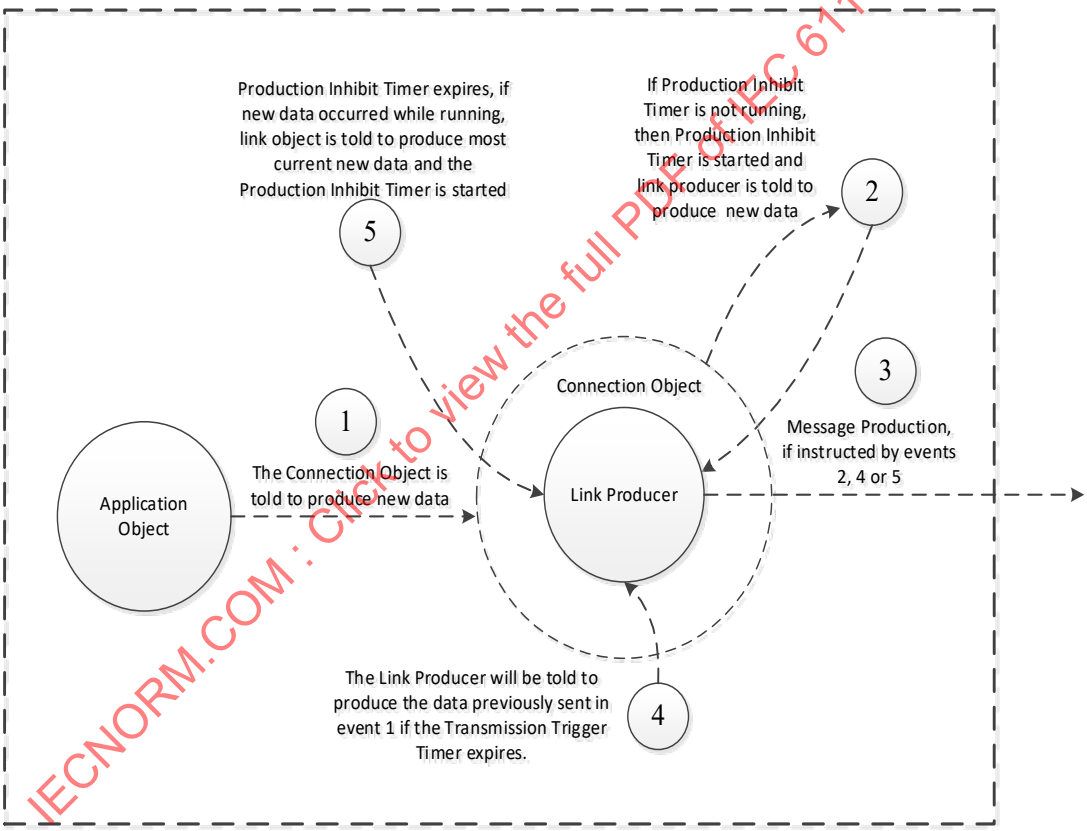


Figure 20 – Production Inhibit Timer behavior

Important: The Production Inhibit Timer is initialized with the value in the **production_inhibit_time** attribute. If the **production_inhibit_time** attribute contains the value zero (0), then the Production Inhibit Timer is not activated and/or used by the Client end-point.

Important: Server end-points do not activate this timer.

6.2.3.3 Connection ASE service specification

6.2.3.3.1 Supported services

Subclause 6.2.3.3 contains the definition of services that are unique to this ASE. The services defined for this ASE are

Connection_Bind

Producing_Application_Lookup

SafetyOpen

SafetyClose.

6.2.3.3.2 Connection service common parameters

The FAL service common parameters specified in 6.2.1.3.2 are also used with Connection services.

6.2.3.3.3 Connection service status codes

Specific status codes for the Connection are detailed in IEC 61158-6-2.

6.2.3.3.4 Connection_Bind

6.2.3.3.4.1 Service overview

The Connection_Bind object specific service binds two dynamically created I/O connection instances together for purposes of connection timeouts and deletions.

6.2.3.3.4.2 Service primitives

The service parameters for this service are shown in Table 60.

Table 60 – Connection_Bind service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Bound Instances	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Bound Instances

Instance numbers of Connection objects to be bound.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.3.3.4.3 Service procedure

No specific service procedure.

6.2.3.3.5 Producing_Application_Lookup

6.2.3.3.5.1 Service overview

The Producing_Application_Lookup object specific service provides a mechanism to find one or more connection instances in the Established state producing data from a given application object.

6.2.3.3.5.2 Service primitives

The service parameters for this service are shown in Table 61.

Table 61 – Service_Name service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Producing Application Path	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Instance Count			M	M(=)
Connection Instance List			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Producing application path

Connection path of producing application to be searched for within connections in the Established state.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Instance count

Number of Instances returned in the Connection Instance List parameter.

Connection instance list

List of instance numbers of connection producing the data from the requested connection path within the node.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 23).

6.2.3.3.5.3 Service procedure

No specific service procedure.

6.2.3.3.6 SafetyOpen

See IEC 61784-3-2 for the definition of this service.

6.2.3.3.7 SafetyClose

See IEC 61784-3-2 for the definition of this service.

6.3 ARs

6.3.1 Overview

6.3.1.1 General

One of the roles of the application layer is to establish and maintain connections. A connection is similar to a telephone circuit. When a call is placed, the telephone system selects a path for your call and sets up each switching station in the route to handle it. As long as the call continues, the resulting virtual circuit remains open, carrying data or voice traffic. In the telephone system, a single call can travel over multiple, different type links, and the format of the data can change from link to link, but the connection appears the same to both the caller and the party called. Sound in one end becomes sound out the other.

A connected message assumes previously negotiated resources and parameters at its source, its destination(s), and any intermediate transit points. These resources are referenced by a unique connection identifier and do not need to be contained in each message, only the connection ID is needed to identify the message and refer to its related parameters, thus giving significant savings in message efficiency.

An unconnected message provides a means to communicate on the local link without previously negotiated resources at the destination so it shall carry full destination ID details, internal data descriptors and full source ID details if a reply is requested. Unconnected messages are used mainly to create connections.

The unconnected service is provided by the Unconnected Message Manager (UCMM). Messages received through the UCMM are forwarded to the Message Router (MR), which direct them to the appropriate internal object for execution. Connections are established by specific unconnected messages sent through the Connection Manager (CM) using UCMM services. Connections may be established either to the Message Router (for messaging purpose), or directly to an application object. Connection target is specified using a connection path, and may be either on the local or a remote link, through several intermediate router nodes. Once a connection is established with an application object, the UCMM, MR and

CM are no longer required, since data will be exchanged directly with the connected objet, based on the corresponding connection ID. Connected messages sent to the Message Router will be forwarded to the appropriate internal object for execution.

The connected and unconnected protocols specified within this part of IEC 61158 are collectively known as the Common Industrial Protocol (CIP).

6.3.1.2 Message Router

The Message Router object provides a messaging interface through which a client may address a service to any object class or instance residing in the device. Every node shall contain instance 1 of the Message Router object.

6.3.1.3 Application connections

6.3.1.3.1 Connections

An application connection is a logical connection between two applications, and is made at the application layer of the communication model. An application connection is built on one or two transport connections. The originating application is responsible for creating and binding the client transport instance to the originating application, and the target application is responsible for creating and binding the server transport instance to the target application. An application connection includes the transport connection which, in turn, includes the producers and consumers involved in the network connection.

Since the unidirectional class 0 and 1 transports require a single network connection, two unidirectional transports can be established within a single connection. The two unidirectional transports do not have to be coordinated at either of the applications, but shall be linked in any of the routers.

The application connection can also include additional parameters or data. A target application, for example, may need information on electronic keying, ownership verification, or the detailed configuration to establish the connection. Such information is sent as a structure in a data segment as part of the connection path. The data segment is forwarded by the intermediate nodes and delivered to the target application as additional segments in the CM_open indication service primitive. The data segment shall be interpreted by the target application.

NOTE The target application can check the module type in the electronic key, check or save the ownership information, and use the configuration information to set up the device. Success and reply information or failure and a reason for the failure can be returned in the message reply. The format of the data segment is a function of the target application.

6.3.1.3.2 Production trigger, transport class and CM_RPI

A client application uses the transport class, trigger type and CM_RPI to determine when to sample and send data.

The system supports three types of triggers for the production of data: cyclic, change of state, and application–triggered. These triggers only apply to the client application.

The triggers are used in conjunction with the transport class and CM_RPI to control when data are sampled and sent on the link. Table 62 shows how these factors relate to each other in determining when data are produced.

Table 62 – How production trigger, transport class, and CM_RPI determine when data is produced

Trigger type	Transport class	When data are sampled and sent	When data are repeated
Cyclic	0 – Null	At the CM_RPI	
	1 – Duplicate Detection	At the CM_RPI	
	2 – Acknowledged	At the CM_RPI	At 1/4 the CM_RPI or faster if not acknowledged
	3 – Verified	At the CM_RPI	At 1/4 the CM_RPI or faster if not verified
Change of State	0 – Null	When data changes	At the CM_RPI
	1 – Duplicate Detection	When data changes	At the CM_RPI
	2 – Acknowledged	When data changes	At 1/4 the CM_RPI or faster until acknowledged, and then at the CM_RPI
	3 – Verified	When data changes	At 1/4 the CM_RPI or faster until verified, and then at the CM_RPI
Application	0 – Null	Determined by application	
	1 – Duplicate Detection	Determined by application	
	2 – Acknowledged	Determined by application	At 1/4 the CM_RPI or faster if not acknowledged
	3 – Verified	Determined by application	At 1/4 the CM_RPI or faster if not verified

6.3.1.3.3 Polling

Although related to triggering, polling is a server–only function. The system supports two types of polling: asynchronous and synchronous.

Asynchronous polling uses transport class 2 (acknowledged). When the server application receives the poll request, the current data buffer is sent as the acknowledgement. This method requires the application to continuously sample data and update the buffer. The application sampling the data does not know which sample is sent in response to [via] a poll request.

The synchronous poll shall use transport class 3 (verified). When the server application is notified of the poll request, it shall sample new data, which it returns as the verification.

6.3.1.4 Transport connections

6.3.1.4.1 General

A transport connection is a logical binding between two transport instances that is made to enable two applications to transfer data over network connections. A transport instance is a specific implementation of a transport function. Every transport connection is built on one or two network connections. Figure 21 uses the communication model to show the context in which transport connections are made.

NOTE 1 Input, Download, Upload and Scanner functions are examples of users of the transport services, they are not described in IEC 61158 series.

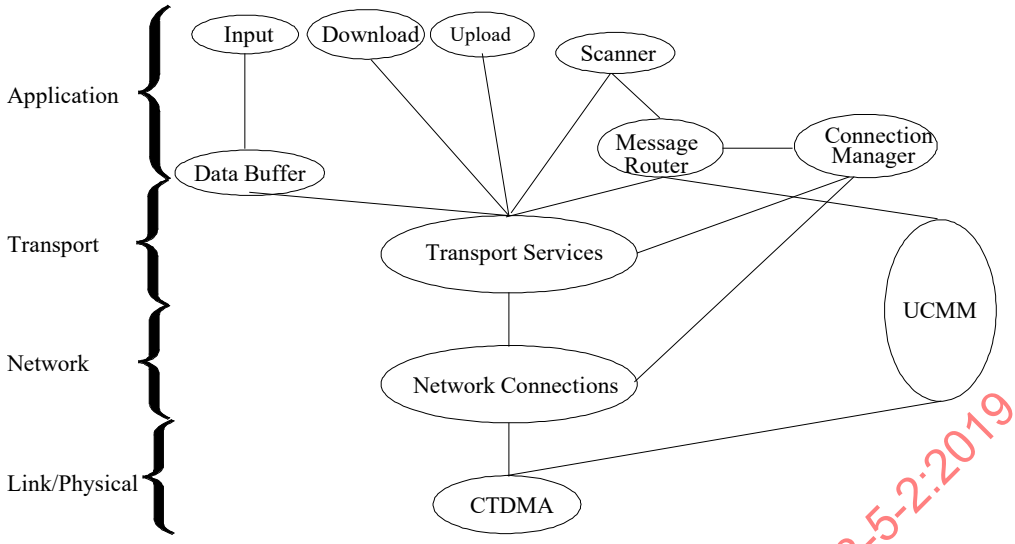


Figure 21 – Context of transport services within the connection model

In Figure 22, application A interfaces to a client transport instance to trigger the production of data over the network connection, and application B interfaces to a server transport instance to be notified that data has been written to the transport protocol data unit (T–PDU) buffer. A T–PDU comprises one packet of data as well as its link, network, and transport headers. A T–PDU buffer is a memory location in which one T–PDU can be stored.

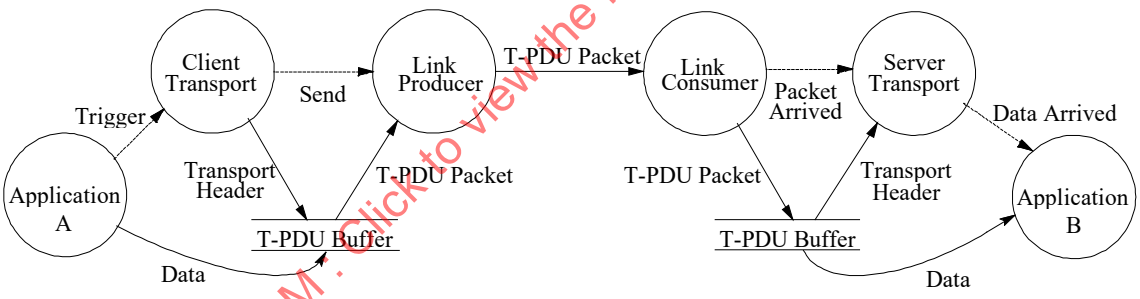


Figure 22 – Application–to–application view of data transfer

Data is not directly written to or read from the transport instances: data is sent to or removed from the T–PDU buffers. The client and server transport instances are responsible only for managing the transport headers of the data packets in the T–PDU buffers. See specification of transport classes in IEC 61158-6-2, 9.3 for additional details on the responsibilities of the client and server transport instances.

NOTE 2 Producers, consumers, and transport instances do not have public interfaces. They are accessible only through the Connection Manager that creates them.

6.3.1.4.2 Components of transport connections

6.3.1.4.2.1 Components

A transport connection includes the transport instance(s), the client– and server–side T–PDU buffer(s), and the network connection(s) involved. The network connection, in turn, includes the link producer(s) and link consumer(s). The originating application is responsible for creating all of the above components, and for creating bindings between those components that result in the desired transport connection being created.

6.3.1.4.2.2 Network connections

A logical binding between a producer and consumer form a network connection. The Connection Manager shall establish this connection. When a producer wants to send a message, it shall provide just the production ID. The production ID implies the path that the Connection Manager specified. Because of this implied route, no addressing or routing information other than the production ID is used at run time.

There is no queuing of data along a network connection. Overwrites are permitted at all intermediate nodes. Those applications that require queuing shall use an appropriate transport class to deliver this behavior.

Link producer

The link producer is responsible for fetching data from the data buffer and producing it on the data link after the send event has been received. Data buffer management schemes such as overwrite, double buffered, triple buffered are not excluded, but since they are implementation specific, they are not specified here.

Behavior of the link producer is shown in Figure 23.

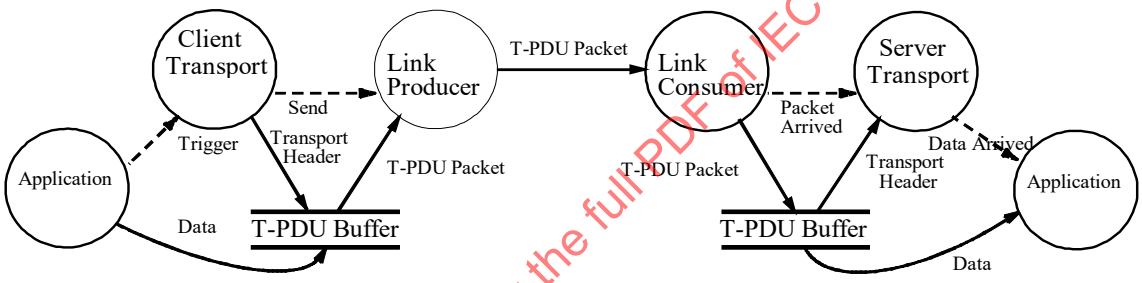


Figure 23 – Data flow diagram for a link producer

Class object description of the Link Producer is detailed below.

FAL ASE: **Link Producer ASE**
CLASS: **Link_Producer**
CLASS ID: **<na>**
PARENT CLASS: **TOP**
STATIC ATTRIBUTES:

1	(m)	Attribute:	Production ID
2	(m)	Attribute:	Priority = URGENT | SCHEDULED | HIGH | LOW
3	(m)	Attribute:	Connection Size
4	(m)	Attribute:	Destination Type = POINT2POINT | MULTIPOINT
5	(m)	Attribute:	State

SERVICES:

1	(m)	OpsService:	Create
2	(o)	OpsService:	Delete
3	(m)	OpsService:	Send

Create

Create instance of a link producer.

Delete

Delete instance of a link producer.

Send

Produce the T-PDU buffer.

Link consumer

The link consumer is responsible for receiving T-PDU packets, storing them in the T-PDU buffer and notifying the server transport class that a packet has arrived. Data buffer management schemes such as overwrite, double buffered, triple buffered are not excluded, but since they are implementation specific, they are not specified here.

Behavior of the link producer is shown in Figure 24.

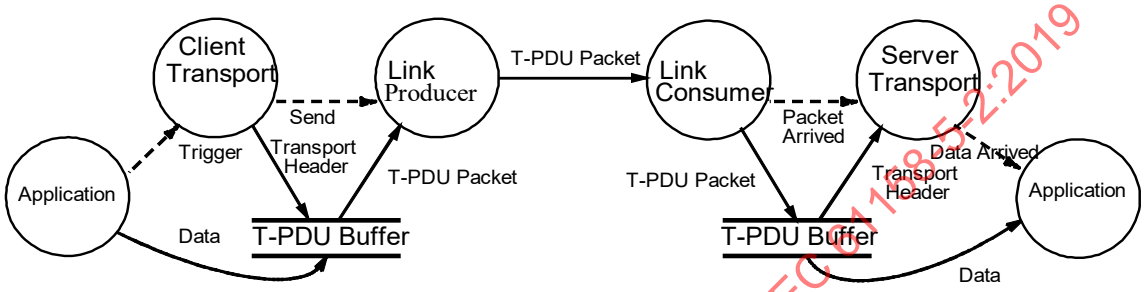


Figure 24 – Data flow diagram for a link consumer

Class object description of the Link Consumer is detailed below.

FAL ASE:			**Link Consumer ASE**
CLASS:			**Link_Consumer**
CLASS ID:			**<na>**
PARENT CLASS:			**TOP**
STATIC ATTRIBUTES:			
1	(m)	Attribute:	Consumption ID
2	(m)	Attribute:	Priority = URGENT | SCHEDULED | HIGH | LOW
3	(m)	Attribute:	Connection Size
4	(m)	Attribute:	Destination Type = POINT2POINT | MULTIPOINT
5	(m)	Attribute:	State
SERVICES:			
1	(m)	OpsService:	Create
2	(o)	OpsService:	Delete
3	(m)	OpsService:	Send

Create

Create instance of a link consumer.

Delete

Delete instance of a link consumer.

6.3.1.4.2.3 Triggers

A trigger shall cause data to be sent. This trigger can be anything that makes sense from an application point of view, such as a clock on a module, a network trigger, a scheduler, an asynchronous event or a timer, as shown in Figure 25.

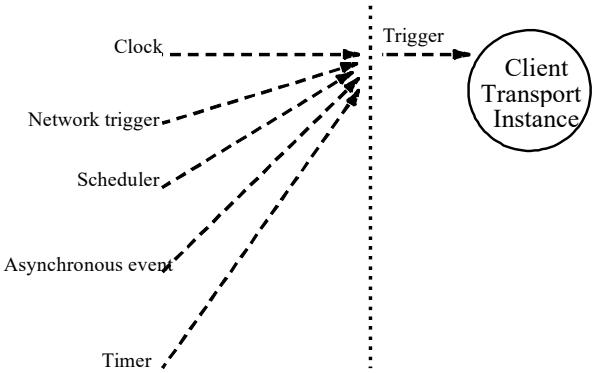


Figure 25 – Triggers

It is important to note that the writing of data to the buffer and triggering of that data are separate functions. An application may write to a buffer more often than it triggers data or trigger data more often than it writes to the buffer. Applications shall write data to the data buffer before it is initially triggered to ensure that valid data is sent.

6.3.1.4.3 Creating transport connections

After the application has bound the producer to the consumer at the network level of the communication model, the application creates the transport connection by binding the producer, consumer, and T–PDU buffer(s) to the transport instances at the transport level. An application may support as many transport connections as resources permit.

The simplest transport functions use a single network connection, which can be either point–to–point or multipoint; the more complex transport functions also use one or more point–to–point return network connections. The transport functions correspond to the transport classes. Each transport class provides a specified level of transport services.

6.3.1.4.4 Binding transports to producers and consumers of network connections

The binding rules followed to create transport connections are as follows.

– A client transport instance is bound to exactly one producer instance.
– A client transport instance can be bound to zero, one, or many consumer instances.
– A server transport instance can be bound to zero or one producer instances.
– A server transport instance is bound to exactly one consumer instance.
– A server transport instance can be bound to a producer instance and a consumer instance, but can not be bound to a producer instance without also being bound to a consumer instance.

The following examples demonstrate application of the binding rules to different kinds of transport connections that include network connections with different destination types.

– For a point–to–point connection that does not have a reverse data path, the producer is bound to the client transport instance and the consumer is bound to the server transport instance as shown in Figure 26:

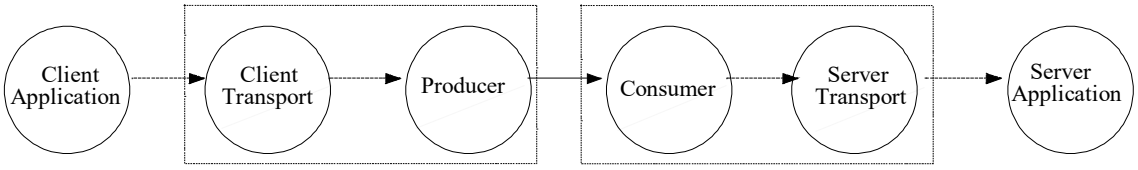
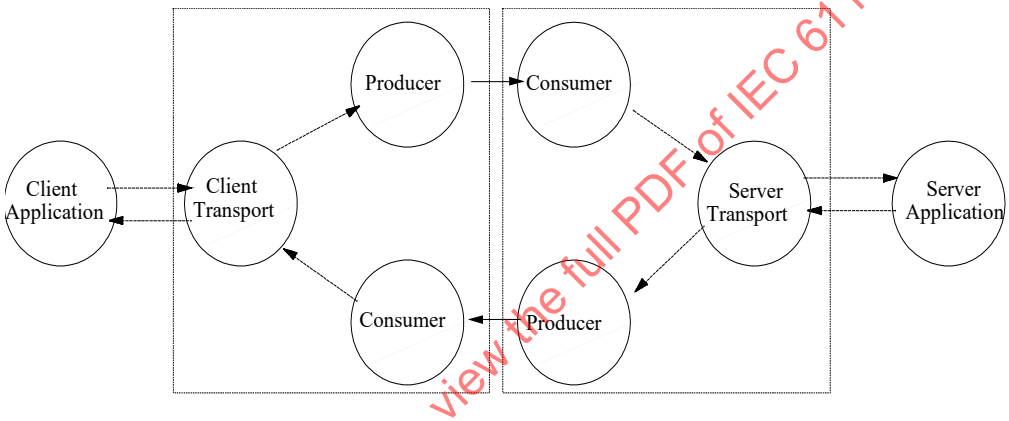


Figure 26 – Binding transport instances to the producer and consumer of a transport connection that does not have a reverse data path

– For a point–to-point connection that does have a reverse data path:

- in the client–to–server direction, the producer is bound to the client transport instance, and the consumer is bound to the server transport instance,
- in the server–to–client direction, the producer is bound to the server transport instance, and the consumer is bound to the client transport instance.

Figure 27 illustrates this scenario.



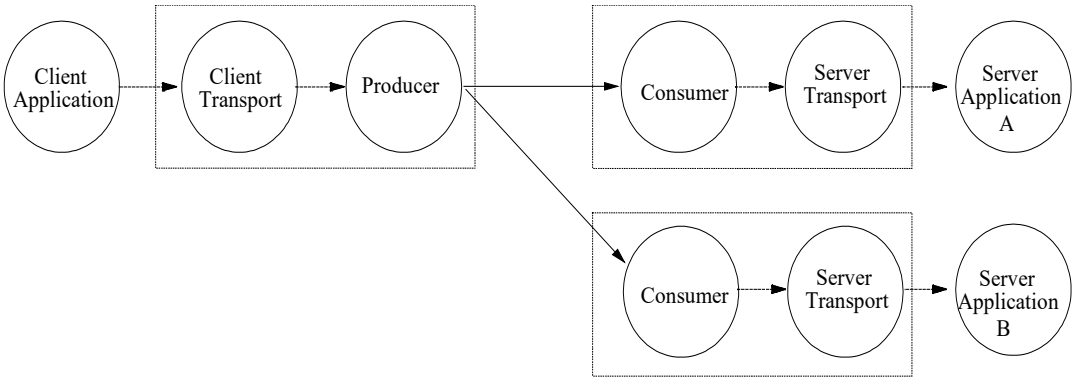
NOTE In this configuration, the transport connection includes:

1) the client and server transport instances;
2) the network connections, including the producers and consumers in both the client-to-server and server-to-client directions.

Figure 27 – Binding transport instances to the producers and consumers of a transport connection that does have a reverse data path

– For a multipoint connection that does not have a reverse data path, the producer is bound to the client transport instance, and each consumer is bound to a separate server transport instance. It is important to note that there is only one client transport instance, and there are as many server transport instances as there are client–to–server network connections.

Figure 28 illustrates this scenario.



NOTE In this configuration, there are two transport connections: one from client transport to producer to consumer A to server transport A, and one from client transport to producer to consumer B to server transport B.

Figure 28 – Binding transport instances to the producer and consumers of a multipoint connection when the transport connection does not have a reverse data path

– For a multipoint connection that does have a reverse data path:

- In the client–to–server direction, the producer is bound to the client transport instance, and each consumer is bound to a separate server transport instance.
- In the server–to–client direction, each producer is bound to a separate server transport instance, and each consumer is bound to the client transport instance. It is important to note that there is only one client transport instance, and there are as many server transport instances as there are client–to–server network connections.

Figure 29 illustrates this scenario.

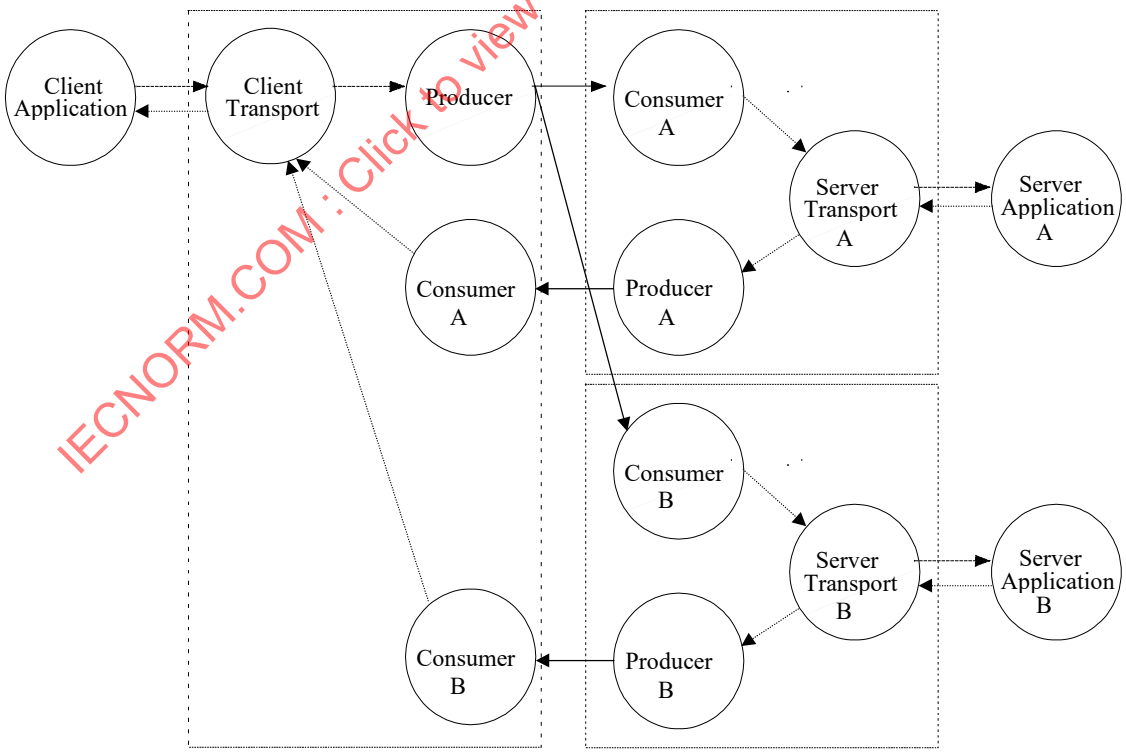


Figure 29 – Binding transport instances to the producers and consumers of a multipoint connection when the transport connection does have reverse data paths

6.3.1.4.5 Application type

6.3.1.4.5.1 General

The application type determines the target behavior concerning the relationship between different connections each sharing a producer, and shall be one of LISTEN_ONLY, INPUT_ONLY, EXCLUSIVE_OWNER or REDUNDANT_OWNER.

The O=>T application path, defined by the target, specifies the application type. In addition the Redundant Owner bit in the O=>T Network Connection Parameters parameter shall be used to differentiate between EXCLUSIVE_OWNER and REDUNDANT_OWNER.

The number of simultaneous connections is implementation dependent. When the target cannot accept more connections it shall respond with appropriate general/extended status.

6.3.1.4.5.2 Listen only

A Listen Only connection type is one that may provide application data in the T=>O direction and a heartbeat in the O=>T direction. If a connection has an application type of Listen Only, it shall be dependent on a non-Listen Only application connection for its existence. If the target receives a request to open a Listen Only connection when there is no existing non-Listen Only connection for the requested T=>O application path, the target shall respond with Non-Listen Only Connection Not Opened (General Status = 0x01, Extended Status = 0x0119).

The O=>T connection shall use the appropriate heartbeat format for the transport class.

If the device also supports an Input Only connection to the same T=>O application path then the Listen Only and Input Only O=>T heartbeat application paths shall be different.

Devices that wish to listen to multicast data without providing configuration may use this application type.

If the last connection on which a Listen Only connection depends is closed or times out, the target device shall stop sending the T=>O data for the Listen Only connection resulting in a time out at the originator device.

6.3.1.4.5.3 Input only

An Input Only connection type is one that may provide application data in the T=>O direction and a heartbeat in the O=>T direction. If a connection has an application type of Input Only, it shall not be dependent on any other connection for its existence.

The O=>T connection shall use the appropriate heartbeat format for the transport class.

If the device also supports a Listen Only connection to the same T=>O application path then the Listen Only and Input Only O=>T heartbeat application paths shall be different.

6.3.1.4.5.4 Exclusive owner

An Exclusive Owner connection type is one that may provide application data in both the T=>O direction and the O=>T direction. If a connection has an application type of Exclusive Owner, it shall not be dependent on any other connection for its existence.

O=>T application data that controls outputs may be present. A target shall only accept one Exclusive Owner connection which specifies the same O=>T application path. In addition, the target may accept listen only and input only connections that use the same multicast T=>O data.

The term connection owner refers to the connection originator whose O=>T packets are being consumed by the target object. The term owning connection refers to the connection associated with the connection owner.

When an Exclusive Owner connection timeout occurs in a target device, the target device shall stop sending the associated T=>O data. The T=>O data shall not be sent even if one or more Input Only connections exist. This requirement is necessary to signal the originator of the Exclusive Owner connection that the O=>T data is no longer being received by the target device.

NOTE One possible way to prevent an Exclusive Owner connection timeout in a target device from stopping the T=>O production for other network connections to the same T=>O application path, is for the target device to support production of the T=>O data as point to point for the Exclusive Owner connection and simultaneously support multicast production for other network connections to the same T=>O application path.

6.3.1.4.5.5 Redundant owner

a) General

A Redundant Owner connection type is one that may provide application data in both the T=>O direction and the O=>T direction. The Redundant Owner connection allows multiple separate originator applications to each establish an independent, identical connection to the transport of a target application. The target transport shall in turn send events to the target application so that the Redundant Owner connection appears as a single, Exclusive Owner connection to the target application. At most one of the originator applications shall have its application data applied to the target application. The target transport shall multicast the target application data to each of the originator applications.

When a single Redundant Owner connection exists and a Redundant Owner connection timeout occurs in a target device, the target device shall stop sending the associated T=>O data. The T=>O data shall not be sent even if one or more input only connections exist. This requirement is necessary to signal the originator of the Redundant Owner connection that the O=>T data is no longer being received by the target device.

NOTE 1 One possible way to prevent a Redundant Owner connection timeout in a target device from stopping the T=>O production for other network connections to the same T=>O application path, is for the target device to support production of the T=>O data as point to point for the Redundant Owner connection and simultaneously support multicast production for other network connections to the same T=>O application path.

NOTE 2 The Redundant Owner connection can commonly be used in applications where up time is at a premium. In these applications, multiple originators can each establish a connection to a target application (typically an output application). If an originator application fails or otherwise gives up control, another originator application can quickly have its application data applied to the target application without having to establish a connection with the target.

b) Establishing the connection

A Redundant Owner connection and an Exclusive Owner connection shall not both be established to a target application at the same time. Multiple Redundant Owner connections may be established to the same target simultaneously provided that the following parameters of the Forward_Open request are identical and the connection paths match: O2T_RPI, O2T_connection_parameters, T2O_RPI, T2O_connection_parameters, xport_type_and_trigger. The connection path shall match including any data segments.

The target shall return a status code of "connection failure" and an extended status of "redundant connection mismatch" if any of these fields do not match.

The target transport shall only send the CM_open indication to the target application when the first CM_Forward_Open PDU is received. Subsequent redundant Forward_Open PDUs shall not cause a CM_open indication to be sent to the target application.

NOTE 3 The Redundant Owner element of the connection parameters in the CM_Open request specifies whether the connection is an Exclusive Owner or Redundant Owner connection (see 6.2.2.3.2).

6.3.2 UCMM AR formal model

6.3.2.1 Description

The UCMM provides an unconnected request/response message service, limited to a single link that supports multiple outstanding messages. The required number of outstanding messages is implementation specific. The UCMM shall be present in all nodes, and shall support at least one outstanding message.

NOTE Most I/O modules need only support a single message while controllers typically support multiple outstanding messages.

Setting up a connection requires communication between devices. This initial communication requires an unconnected message service. The UCMM, which only operates on a single link, is different from other transports. That is, while connected transport types use a single buffer to store packets, with new packets overwriting old ones, UCMM packets are queued at each node instead. Multiple connections can, therefore, be established at the same time. This reduces the overall time needed to set up all the connections in a system.

All incoming UCMM requests shall be internally forwarded to the Message Router, who is responsible for sending them to the target object instance or element, based on the path specified in the request.

6.3.2.2 Formal class definition

FAL ASE:			**AR ASE**
CLASS:			**UCMM_Transaction_Record**
CLASS ID:			**<na>**
PARENT CLASS:			**TOP**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Record Number
STATIC ATTRIBUTES:			
1	(m)	Attribute:	Fixed Tag
2	(m)	Attribute:	Max_Retry
3	(m)	Attribute:	Retry Timeout
DYNAMIC ATTRIBUTES:			
1	(m)	Attribute:	Record State
2	(m)	Attribute:	Sequence Number
3	(m)	Attribute:	Retry Count
4	(m)	Attribute:	Response Timeout
SERVICES:			
1	(m)	OpsService:	UCMM_Create
2	(o)	OpsService:	UCMM_ Delete
3	(m)	OpsService:	UCMM_ Write
4	(m)	OpsService:	UCMM_ Abort

6.3.2.3 Key attributes

Record number

Unique number associated with a given UCMM transaction record.

6.3.2.4 Static attributes

These attributes have fixed values once the Transaction Record is created.

Fixed tag

DLL fixed tag service used to exchange messages on this transaction record

Max_Retry

Maximum number of retries for this transaction record.

Retry timeout

If no response or acknowledgement has been received back after the transmission of a UCMM packet (request or response) within the delay specified by the Retry Timeout, transmission of the packet shall be repeated. The Retry Timeout value shall be fixed for the link at a value which guarantees that both the client and server nodes have an opportunity to transmit an unscheduled DLL packet (see IEC 61158-3-2).

6.3.2.5 Dynamic attributes

These attributes change values based on the current operating state of the Transaction Record.

Record state

Current state of the Transaction Record. This shall be one of INACTIVE, RUNNING or WAITING-FOR-RESPONSE.

Sequence number

Sequence Number is used for duplicate detection. It shall be incremented for every unique message on a given Transaction Record ; however it shall not be incremented on a retry.

The Sequence Number shall be set to zero at initialization. The sequence number value shall be maintained in the INACTIVE state, to be used when the record transitions to running.

Retry count

Current number of retries for a given message. When Retry count reaches Max_Retry, the current transaction shall be automatically aborted.

Response timeout

The Response Timeout attribute indicates the maximum duration of the transaction in microseconds. If no response is received before the time-out expires, the transaction shall be automatically aborted. This value is provided when the UCMM_Write service is invoked.

6.3.2.6 Services

UCMM_Create

The UCMM_Create service attempts to create a UCMM transaction record.

UCMM_ Delete

The UCMM_Delete service deletes an existing transaction record.

UCMM_ Write

The UCMM_Write service is used to send a message over an existing (already created) transaction record.

UCMM_ Abort

The UCMM_Abort service is used either by the client or the server to abort a transaction.

6.3.3 Transport AR formal model

6.3.3.1 Overview

The Transport AR's provides means of exchanging data along previously made connections. Depending on the functions performed by the connection, the Transport AR's fit into one of seven classes.

Applications interface to transport services through the supported transport "classes". Table 63 lists the general–purpose transport classes that have been defined for the systems. Each transport class provides a different level of functionality. Transport "class 0" provides the minimum functionality required of a transport class, enabling data transfer between applications.

Table 63 – Transport classes

Class Number	Class name
0	Null (or Base)
1	Duplicate Detection
2	Acknowledged
3	Verified
4	Non-blocking
5	Non-blocking, Fragmenting
6	Multipoint, Fragmenting

6.3.3.2 Generic transport AR

6.3.3.2.1 Formal class definition

This is the general model for all transport classes.

FAL ASE: **AR ASE**
CLASS: **Generic_Transport**
CLASS ID: **NULL**
PARENT CLASS: **TOP**
ACCESS ATTRIBUTES:

1	(m)	Key Attribute:	Transport identifier

STATIC ATTRIBUTES:

1	(m)	Attribute:	Transport class
2	(o)	Attribute:	Max_Retry
3	(o)	Attribute:	Retry Timeout

DYNAMIC ATTRIBUTES:

1	(m)	Attribute:	Transport State
2	(o)	Attribute:	Sequence Count/Number
3	(o)	Attribute:	Previous Sequence Number
4	(o)	Attribute:	Command
5	(o)	Attribute:	Previous Command
6	(o)	Attribute:	Retry Count
7	(o)	Attribute:	Response Timeout

SERVICES:

1	(o)	OpsService:	TR_Write
2	(o)	OpsService:	TR_Trigger
3	(o)	OpsService:	TR_Packet_arrived
4	(o)	OpsService:	TR_Ack_received

5	(o)	OpsService:	TR_Verify
6	(o)	OpsService:	TR_Status_updated

6.3.3.2.2 Key attribute

Transport identifier

The Transport identifier is generated locally by the originator Connection Manager upon connection establishment, and serves as a further reference for the user to the established connection, i.e. this transport instance.

6.3.3.2.3 Static attributes

These attributes have fixed values once the transport has been created as part of the connection establishment process. They are configured by the Connection Manager. These attributes are defined for management. It is not required for them to be visible to the communications system as accessible attributes.

Trans_class

The Trans_class attribute specifies the transport class selected, and shall be one of NULL (also called "class 0" in both this part of IEC 61158 and IEC 61158-6-2), DUPLICATE_DETECTION ("class 1"), ACKNOWLEDGED ("class 2"), VERIFIED ("class 3"), NON-BLOCKING ("class 4"), NON-BLOCKING_FRAGMENTING ("class 5"), or MULTIPOINT_FRAGMENTING ("class 6").

Max_Retry

Maximum number of retries for this transport instance.

Retry timeout

If no response or acknowledgement has been received back after the transmission of a transport packet within the delay specified by the Retry Timeout, transmission of the packet shall be repeated.

6.3.3.2.4 Dynamic attributes

These attributes change values based on the current operating state of the transport instance.

Transport state

Current state of the transport instance. This shall be one of *NON-EXISTENT, IDLE, READY-TO-RUN, RUNNING, WAIT-FOR-ACKNOWLEDGE or WAIT-FOR-VERIFY.*

Sequence count/sequence number

Sequence Count/Number is used for duplicate detection. It shall be incremented for every unique message on a given transport instance ; however it shall not be incremented on a retry.

The Sequence Count/Number shall be set to zero at initialization.

Previous sequence number

This attribute saves the Sequence Count/Number used by the previous message on this transport instance (used for duplicate detection).

Command

This attribute identifies the type of message sent on this transport instance, and shall be one of:

– NULL, ONLY, FIRST, MIDDLE, LAST for messages from the sender,

– ACK, NAK, NOT-INITIALISED for messages from the receiver.

Previous command

This parameter saves the Command value used by the previous message on this transport instance.

Retry count

Current number of retries for a given message. When Retry count reaches Max_Retry, the current transaction shall be aborted.

Response timeout

The Response Timeout attribute indicates the maximum duration of the transaction. If no response is received before the time-out expires, the transaction shall be aborted.

6.3.3.2.5 Services

TR_Write

The TR_Write service is used by a Client/Sender or Server application to put application data into a Transport PDU buffer (transport classes 0 to 3) or queue (transport classes 4 to 6) for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from a Client/Sender application. It shall be invoked when the data in the T-PDU buffer is to be sent.

TR_Packet_arrived

This service is invoked at a Server/Receiver application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer or queue.

TR_Ack_received

This service is invoked at a Client application to indicate that an acknowledgement has been received. New data (sent together with the acknowledgement) may be available in the T-PDU buffer.

TR_Verify

The TR_Verify service is used by a Server application to send back a verification response, after reception and processing of a request message from a Client application. This verification response may or may not contain data for the Client. This service is also invoked at a Client application to indicate that a verification has been received, and that new data (sent together with the verification) may be available in the T-PDU buffer.

TR_Status_updated

This service is invoked at a Sender application to indicate that status (success/error) for a previous request has been received, and is available in the dedicated Status queue.

NOTE As for all objects, the services Create, Delete, Start, Stop and Reset are available for these transport AR's, but they are not exposed here since they are not directly accessible from the user application (only from the Connection Manager).

6.3.3.3 Class 0 transport ARs

6.3.3.3.1 Description

Transport class 0 (Null transport type) is the simplest transport class, and can use either a point–to–point or multipoint connection. Since transport class 0 does not detect duplicate data packets, the target application is responsible for detecting them.

Class 0 is typically used to transmit or receive inputs on a multipoint connection or outputs on a point–to–point connection. Possible uses of transport class 0 include sampled inputs, standard outputs, cyclic block transfers, diagnostic events, and communication between controllers and operator interface devices.

6.3.3.3.2 Class 0 client transport AR

6.3.3.3.2.1 Formal class definition

This is the definition of the Class 0 Client Transport (Null transport type).

FAL ASE:			**AR ASE**
CLASS:			**Transport_Client_0**
CLASS ID:			**<na>**
PARENT CLASS:			**Generic_Transport**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Transport identifier
STATIC ATTRIBUTES:			
1	(m)	Attribute:	Transport class = NULL
DYNAMIC ATTRIBUTES:			
1	(m)	Attribute:	Transport State = NON-EXISTENT | IDLE | RUNNING
SERVICES:			
1	(m)	OpsService:	TR_Write
2	(m)	OpsService:	TR_Trigger

6.3.3.3.2.2 Attributes

No specific requirement for this object.

6.3.3.3.2.3 Services

TR_Write

The TR_Write service is used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

6.3.3.3.3 Class 0 server transport AR

6.3.3.3.3.1 Formal class definition

This is the definition of the Class 0 Server Transport (Null transport type).

FAL ASE:			**AR ASE**
CLASS:			**Transport_Server_0**
CLASS ID:			**<na>**
PARENT CLASS:			**Generic_Transport**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Transport identifier
STATIC ATTRIBUTES:			
1	(m)	Attribute:	Transport class
DYNAMIC ATTRIBUTES:			
1	(m)	Attribute:	Transport State = NON-EXISTENT | IDLE | RUNNING
SERVICES:			
3	(m)	OpsService:	TR_Packet_arrived

6.3.3.3.3.2 Attributes

No specific requirement for this object.

6.3.3.3.3.3 Services

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, but it will not detect duplicate packets in this case. The new packet is available in the T-PDU buffer.

6.3.3.4 Class 1 transport ARs

6.3.3.4.1 Description

Transport class 1 (Duplicate Detection transport type), like class 0, allows either a point–to–point or multipoint connection. Unlike class 0, the class 1 transport allows to detect the delivery of duplicate data packets.

NOTE Class 1 is preferred to class 0 for targets because the target application does not have to perform duplicate detection, which reduces the overhead required in those end nodes that support the change of state transport trigger. Class 1 allows an efficient change of state trigger, since this class was designed to allow change of state data transfers.

Applications could use this transport class to distinguish between new samples and old samples that were sent to maintain the connection. To maintain the connection, a timer may be used to trigger the production of the current data in the T-PDU buffer. This may allow applications to reduce their overhead by only processing new data samples.

Possible uses of transport class 1 include sampled inputs, standard outputs, cyclic block transfers, diagnostic events, and communication between controllers and operator interface devices.

6.3.3.4.2 Class 1 client transport AR

6.3.3.4.2.1 Formal class definition

This is the definition of the Class 1 Client Transport (Duplicate Detection transport type).

FAL ASE: **AR ASE**
CLASS: **Transport_Client_1**
CLASS ID: **<na>**
PARENT CLASS: **Generic_Transport**
ACCESS ATTRIBUTES:
1 (m) Key Attribute: Transport identifier
STATIC ATTRIBUTES:
1 (m) Attribute: Transport class = DUPLICATE_DETECTION
DYNAMIC ATTRIBUTES:
1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | RUNNING
2 (m) Attribute: Sequence Count/Number
SERVICES:
1 (m) OpsService: TR_Write
2 (m) OpsService: TR_Trigger

6.3.3.4.2.2 Attributes

No specific requirement for this object.

6.3.3.4.2.3 Services

TR_Write

The TR_Write service is used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

6.3.3.4.3 Class 1 server transport AR

6.3.3.4.3.1 Formal class definition

This is the definition of the Class 1 Server Transport (Duplicate Detection transport type).

FAL ASE: **AR ASE**
CLASS: **Transport_ Server_1**
CLASS ID: **<na>**
PARENT CLASS: **Generic_Transport**
ACCESS ATTRIBUTES:
1 (m) Key Attribute: Transport identifier
STATIC ATTRIBUTES:
1 (m) Attribute: Transport class = DUPLICATE_DETECTION
DYNAMIC ATTRIBUTES:
1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | READY-TO-RUN | RUNNING
2 (m) Attribute: Sequence Count/Number
3 (m) Attribute: Previous Sequence Number
SERVICES:
3 (m) OpsService: TR_Packet_arrived

6.3.3.4.3.2 Attributes

No specific requirement for this object.

6.3.3.4.3.3 Services

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer.

6.3.3.5 Class 2 transport ARs

6.3.3.5.1 Description

Transport class 2 (Acknowledged transport type) starts with the behavior in class 1 and adds a return connection that acknowledges the delivery of a packet. The producer connection from the client node can be point to point or multipoint. The connection from the server to the client is point to point.

This transport class allows the application to be notified that the server has received the transmitted packet. This acknowledgement tells the client application only that the data arrived. This does not mean that the server application has read the buffer or that a new sample can be sent without overwriting the buffer.

The main advantage of this class over class 3 (Verified) is that the acknowledgement is returned immediately after the packet arrives at the consumer, while the verify class requires the application to initiate the return verification. The delays associated with receiving the acknowledgement are small, approximately twice the time that is required for a one way transmission.

Transport class 2 uses two connections: the originator–to–target connection, which can be either point–to–point or multipoint; and the return connection, which shall be point–to–point. A multipoint connection using transport class 2 shall have a point–to–point return connection corresponding to each client–to–server connection. The acknowledgement notifies the client transport instance that the consuming node of the network connection has received the packet.

If a server transport instance does not acknowledge receipt of the packet, the client transport instance triggers the client application to retransmit the most recently sent packet; if the client–to–server connection is one of the network connections of a multipoint connection, data is resent over all of the network connections of the multipoint connection. The server application can return data to the client application along with its acknowledgement.

Possible uses of transport class 2 include master/slave applications and communication between controllers and operator interface devices.

6.3.3.5.2 Class 2 client transport AR

6.3.3.5.2.1 Formal class definition

This is the definition of the Class 2 Client Transport (Acknowledged transport type).

FAL ASE:			**AR ASE**
CLASS:			**Transport_Client_2**
CLASS ID:			**<na>**
PARENT CLASS:			**Generic_Transport**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Transport identifier
STATIC ATTRIBUTES:			
1	(m)	Attribute:	Transport class = ACKNOWLEDGED
2	(m)	Attribute:	Max_Retry

3 (m) Attribute: Retry Timeout

DYNAMIC ATTRIBUTES:

1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | RUNNING | WAIT-FOR-ACKNOWLEDGE

2 (m) Attribute: Sequence Count/Number

6 (m) Attribute: Retry Count

7 (m) Attribute: Response Timeout

SERVICES:

1 (m) OpsService: TR_Write

2 (m) OpsService: TR_Trigger

4 (m) OpsService: TR_Ack_received

6.3.3.5.2.2 Attributes

No specific requirement for this object.

6.3.3.5.2.3 Services

TR_Write

The TR_Write service is used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

TR_Ack_received

This service is invoked at the Client application to indicate that an acknowledgement has been received. New data (sent together with the acknowledgement) may be available in the T-PDU buffer.

6.3.3.5.3 Class 2 server transport AR

6.3.3.5.3.1 Formal class definition

This is the definition of the Class 2 Server Transport (Acknowledged transport type).

FAL ASE: **AR ASE**

CLASS: **Transport_ Server_2**

CLASS ID: **<na>**

PARENT CLASS: **Generic_Transport**

ACCESS ATTRIBUTES:

1 (m) Key Attribute: Transport identifier

STATIC ATTRIBUTES:

1 (m) Attribute: Transport class = ACKNOWLEDGED

DYNAMIC ATTRIBUTES:

1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | READY-TO-RUN | RUNNING

2 (m) Attribute: Sequence Count/Number

3 (m) Attribute: Previous Sequence Number

SERVICES:

1 (m) OpsService: TR_Write

3 (m) OpsService: TR_Packet_arrived

6.3.3.5.3.2 Attributes

No specific requirement for this object.

6.3.3.5.3.3 Services

TR_Write

The TR_Write service is used by the Server application to put application data into a Transport PDU buffer for further transmission together with the acknowledgement.

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer.

6.3.3.6 Class 3 transport ARs

6.3.3.6.1 Description

Transport class 3 (Verified transport type) starts with the behavior in class 1 and adds a return connection that verifies that the application has received the packet. The sequence count that shall be returned with the verify is the same as the sequence count sent with the data. This return connection is used to identify the verification message.

This transport class allows the application to be notified that the server application has responded to the transmitted packet. This verification tells the client application not only that the data arrived but that the server application has read the buffer and that a new sample can be sent without overwriting the buffer.

The main advantage of this class over class 2 (Acknowledgement) is that verification conveys an application response. The disadvantage is that the delay in receiving the verification can be very difficult to predict.

Transport class 3 uses two connections: the originator–to–target connection, which can be either point–to–point or multipoint; and the return connection, which shall be point–to–point. A multipoint connection using transport class 3 shall have a point–to–point return connection corresponding to each client–to–server connection. The verification notifies the client application that the server application has received and read the transmitted data. If a server application does not verify receipt of the packet, the client application retransmits the most recently sent packet; if the client–to–server connection is a multipoint one, the data is retransmitted over all of the network connections of the multipoint connection.

Possible uses of transport class 3 include change–of–state inputs and outputs, asynchronous [independent] block transfers, event acknowledgements, communication between controllers and operator interface devices, uploads, downloads, and messaging.

6.3.3.6.2 Class 3 client transport AR

6.3.3.6.2.1 Formal class definition

This is the definition of the Class 3 Client Transport (Verified transport type).

FAL ASE:	**AR ASE**
CLASS:	**Transport_Client_3**
CLASS ID:	**<na>**
PARENT CLASS:	**Generic_Transport**

ACCESS ATTRIBUTES:

1	(m)	Key Attribute:	Transport identifier

STATIC ATTRIBUTES:

1	(m)	Attribute:	Transport class = VERIFIED
2	(m)	Attribute:	Max_Retry
3	(m)	Attribute:	Retry Timeout

DYNAMIC ATTRIBUTES:

1	(m)	Attribute:	Transport State = NON-EXISTENT | IDLE | RUNNING | WAIT-FOR-VERIFY
2	(m)	Attribute:	Sequence Count/Number
6	(m)	Attribute:	Retry Count
7	(m)	Attribute:	Response Timeout

SERVICES:

1	(m)	OpsService:	TR_Write
2	(m)	OpsService:	TR_Trigger
5	(m)	OpsService:	TR_Verify

6.3.3.6.2.2 Attributes

No specific requirement for this object.

6.3.3.6.2.3 Services

TR_Write

The TR_Write service is used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

TR_Verify

This service is invoked at the Client application to indicate that a verification has been received, and that new data (sent together with the verification) may be available in the T-PDU buffer.

6.3.3.6.3 Class 3 server transport AR

6.3.3.6.3.1 Formal class definition

This is the definition of the Class 3 Server Transport (Verified transport type).

FAL ASE:	**AR ASE**
CLASS:	**Transport_Server_3**
CLASS ID:	**<na>**
PARENT CLASS:	**Generic_Transport**

ACCESS ATTRIBUTES:

1	(m)	Key Attribute:	Transport identifier

STATIC ATTRIBUTES:

1	(m)	Attribute:	Transport class = VERIFIED

DYNAMIC ATTRIBUTES:

1	(m)	Attribute:	Transport State = NON-EXISTENT | IDLE | READY-TO-RUN | WAIT-FOR-VERIFY | RUNNING
2	(m)	Attribute:	Sequence Count/Number
3	(m)	Attribute:	Previous Sequence Number

SERVICES:

1	(m)	OpsService:	TR_Write

3	(m)	OpsService:	TR_Packet_arrived
5	(m)	OpsService:	TR_Verify

6.3.3.6.3.2 Attributes

No specific requirement for this object.

6.3.3.6.3.3 Services

TR_Write

The TR_Write service is used by the Server application to put application data into a Transport PDU buffer for further transmission together with the verification.

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer.

TR_Verify

The TR_Verify service is used by the Server application to send back a verification response, after reception and processing of a request message from the Client application. This verification response may or may not contain data for the Client.

6.3.3.7 Class 4 transport ARs

NOTE Subclause contents from the previous edition have been obsoleted.

6.3.3.8 Class 5 transport ARs

NOTE Subclause contents from the previous edition have been obsoleted.

6.3.3.9 Class 6 transport ARs

NOTE Subclause contents from the previous edition have been obsoleted.

6.3.4 AR ASE services

6.3.4.1 Supported services

Subclause 6.3.4 contains the definition of services that are unique to this ASE. The services defined for this ASE are:

- UCMM_Create
- UCMM_ Delete
- UCMM_ Write
- UCMM_ Abort
- TR_Write
- TR_Trigger
- TR_Packet_arrived
- TR_Ack_received
- TR_Verify
- TR_Status_updated

6.3.4.2 Common parameters

Record number

This parameter is the unique number associated with the UCMM transaction record to be used for the current transaction.

Transport identifier

The Transport identifier is generated locally by the originator Connection Manager upon connection establishment, and serves as a further reference for the user to the established connection, i.e. this transport instance.

6.3.4.3 UCMM_Create

6.3.4.3.1 Service overview

The UCMM_Create service request attempts to create a UCMM transaction record.

6.3.4.3.2 Service primitives

The service parameters for this service are shown in Table 64.

Table 64 – UCMM_Create service parameters

Parameter name	Req	Cnf
Argument	M	
Fixed Tag	M	
Max_Retry	M	
Result		M
Record Number		M
Service Status		M

Argument

The argument contains the parameters of the service request.

Fixed Tag

DLL fixed tag service used to exchange messages on this transaction record (regular UCMM or management UCMM).

Max_Retry

Maximum number of retries for this transaction record.

Result

This selection type parameter indicates whether the service request succeeded or failed.

Record number

This parameter is the unique number returned by the UCMM and associated with the newly created UCMM transaction record.

Service Status

This parameter provides information on the result of service execution. It shall be one of SUCCESS or CANNOT_CREATE.

6.3.4.3.3 Service procedure

The UCMM_Create service request from the user is processed locally by the UCMM. This service may also be invoked by internal objects (e.g. the Connection Manager).

A transaction record shall be created before any message can be exchanged through the UCMM. Only one message shall be outstanding for a given transaction record. If multiple outstanding messages are required by the user application, then multiple transaction records are required.

6.3.4.4 UCMM_Delete

6.3.4.4.1 Service overview

The UCMM_Delete service deletes an existing transaction record.

6.3.4.4.2 Service primitives

The service parameters for this service are shown in Table 65.

Table 65 – UCMM_Delete service parameters

Parameter name	Req
Argument	M
Record Number	M

Argument

The argument contains the parameters of the service request.

Record number

This parameter is the unique number associated with the UCMM transaction record to be deleted.

6.3.4.4.3 Service procedure

The UCMM_Delete service request from the user is processed locally by the UCMM. This service may also be invoked by internal objects (e.g. the Connection Manager).

6.3.4.5 UCMM_Write

6.3.4.5.1 Service overview

The UCMM_Write service is used to send a message over an existing (already created) transaction record.

6.3.4.5.2 Service primitives

The service parameters for this service are shown in Table 66.

Table 66 – UCMM_Write service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M		
Record Number	M	M(=)		
Dest_ID	M			
Source_ID		M		
UCMM_service	M			
Response Timeout	M			
Transaction_Priority	M			
App_Data	M	M(=)		
Result			M	M
Record Number			M	M(=)
Service Status				M
App_Data			M	M(=)

Argument

The argument contains the parameters of the service request.

Dest_ID

This is the MAC ID of the target (server) device for this transaction.

Source_ID

This is the MAC ID of the originating (client) device for this transaction.

UCMM_Service

The UCMM_Service parameter specifies the delivery mechanism to use for this message and shall be one of:

– Request_With_Acknowledge (request with retry until acknowledged)
– Request_With_No_Response (request with no acknowledgement and no response)
– Request_With_No_ACK (request with retry until response – no acknowledgement)
– Request_With_No_Retry (request with no retry, and response with no retry)

NOTE 1 These values correspond to request command codes contained in the UCMM header (see IEC 61158-6-2).

Response timeout

The Response Timeout attribute indicates the maximum duration of the transaction in microseconds. If no response is received before the time-out expires, the transaction shall be automatically aborted, and the Service Status parameter of the confirmation shall be TIMEOUT.

Transaction_Priority

The Transaction_priority parameter determines which data-link layer priority to use for the messages, and shall be one of LOW or HIGH.

NOTE 2 The definition of DLL priority is specified in IEC 61158-4-2.

App_Data

The App_Data parameter contains application data (service request parameters). The corresponding data length shall be less than 504 octets since any UCMM packet shall fit into a single Lpacket.

Result

This selection type parameter indicates whether the service request succeeded or failed.

Service status

This parameter provides information on the result of service execution. It shall be one of:

SUCCESS

RECORD_NOT_CREATED

PACKET_TOO_BIG

NO_ACKNOWLEDGE

ABORTED

TIMEOUT

App_Data

The App_Data parameter contains application data (service response parameters). The corresponding data length shall be less than 504 octets since any UCMM packet shall fit into a single Lpacket.

6.3.4.5.3 Service procedure

The UCMM_Write request shall send a message to the device on the local link specified by the Dest_ID parameter. When the message arrives at the target device, this shall result in a UCMM_Write indication at the Message Router in that device (the Message Router shall receive all UCMM_Write indications). The Message Router, in turn, shall forward the message to the appropriate target object. When the object responds, the Message Router shall forward the response via the UCMM_Write response primitive. When the response arrives at the device which originated the request, a UCMM_Write confirmation shall be returned to the application object which sent the request.

6.3.4.6 UCMM_Abort

6.3.4.6.1 Service overview

The UCMM_Abort service is used either by the client or the server to abort a transaction.

6.3.4.6.2 Service primitives

The service parameters for this service are shown in Table 67.

Table 67 – UCMM_Abort service parameters

Parameter name	Req
Argument	M
Record Number	M

Argument

The argument contains the parameters of the service request.

Record number

This parameter is the unique number associated with the UCMM transaction record for which the pending transaction is to be aborted.

6.3.4.6.3 Service procedure

The UCMM_Abort service request from the user is processed locally by the UCMM. This service may also be invoked by internal objects.

When called by the client, the UCMM_Abort service shall abort the transaction corresponding to the specified transaction record, which shall remove the packet from the local DLL queue if it has not yet been transmitted. When called by the server, this request shall abort the transaction, which shall suppress the response from being sent. This service need not have a corresponding confirmation since the confirmation of the original request shall serve this purpose by returning a status of ABORTED.

6.3.4.7 TR_Write

6.3.4.7.1 Service overview

The TR_Write service is used by a Client/Sender or Server application to put application data into a Transport PDU buffer (transport classes 0 to 3) or queue (transport classes 4 to 6) for further transmission. The data written into the Transport PDU buffer or queue has no fixed specification or format.

NOTE Usage of PDU buffers and queues is described in IEC 61158-6-2.

6.3.4.7.2 Service primitives

The service parameters for this service are shown in Table 68.

Table 68 – TR_Write service parameters

Parameter name	Req
Argument	M
Transport identifier	M
App_Data	M

Argument

The argument contains the parameters of the service request.

App_Data

The App_Data parameter contains application data (service request/response parameters or user formatted data).

6.3.4.7.3 Service procedure

Since they are dependant on the transport class selected, service procedures for this service are detailed in IEC 61158-6-2.

6.3.4.8 TR_Trigger

6.3.4.8.1 Service overview

The TR_Trigger service initiates the actual transfer of Transport PDU from a Client/Sender application. It shall be invoked when the data in the T-PDU buffer is to be sent. Depending on the type of trigger selected at connection establishment for this transport instance, this service can be requested either directly by the user application, or internally by a timer associated with the connection.

6.3.4.8.2 Service primitives

The service parameters for this service are shown in Table 69.

Table 69 – TR_Trigger service parameters

Parameter name	Req
Argument	M
Transport identifier	M

Argument

The argument contains the parameters of the service request.

6.3.4.8.3 Service procedure

Since they are dependant on the transport class selected, service procedures for this service are detailed in IEC 61158-6-2.

6.3.4.9 TR_Packet_arrived

6.3.4.9.1 Service overview

This service is invoked at a Server/Receiver application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer or queue.

6.3.4.9.2 Service primitives

The service parameters for this service are shown in Table 70.

Table 70 – TR_Packet_arrived service parameters

Parameter name	Ind
Argument	M
Transport identifier	M
Data_Arrived	M
Duplicate_Arrived	M

Argument

The argument contains the parameters of the service indication.

Data_ Arrived / Duplicate_Arrived

The Data_Arrived and Duplicate_Arrived parameters indicate whether the new packet contains new data, or whether it is just a duplicate from a previously received packet.

NOTE A Duplicate Arrived event is normally not of interest to the application; however, it could be useful in triggering a watchdog timer to indicate that the client's application is alive and triggering.

6.3.4.9.3 Service procedure

Since they are dependant on the transport class selected, service procedures for this service are detailed in IEC 61158-6-2.

6.3.4.10 TR_Ack_received

6.3.4.10.1 Service overview

This service is invoked at a Client application to indicate that an acknowledgement has been received. New data (sent together with the acknowledgement) may be available in the T-PDU buffer.

6.3.4.10.2 Service primitives

The service parameters for this service are shown in Table 71.

Table 71 – TR_Ack_received service parameters

Parameter name	Cnf
Result	M
Transport identifier	M

Result

This parameter provides information for the receiving application.

6.3.4.10.3 Service procedure

Since they are dependant on the transport class selected, service procedures for this service are detailed in IEC 61158-6-2.

6.3.4.11 TR_Verify

6.3.4.11.1 Service overview

The TR_Verify service is used by a Server application to send back a verification response, after reception and processing of a request message from a Client application. This verification response may or may not contain data for the Client. This service is also invoked at a Client application to indicate that a verification has been received, and that new data (sent together with the verification) may be available in the T-PDU buffer.

6.3.4.11.2 Service primitives

The service parameters for this service are shown in Table 72.

Table 72 – TR_Verify service parameters

Parameter name	Rsp	Cnf
Result	M	M
Transport identifier	M	M

Result

This parameter provides information for the receiving application.

6.3.4.11.3 Service procedure

Since they are dependant on the transport class selected, service procedures for this service are detailed in IEC 61158-6-2.

6.3.4.12 TR_Status_updated

6.3.4.12.1 Service overview

This service is invoked at a Sender application to indicate that status (success/error) for a previous request has been received, and is available in the dedicated Status queue.

6.3.4.12.2 Service primitives

The service parameters for this service are shown in Table 73.

Table 73 – TR_Status_updated service parameters

Parameter name	Cnf
Result	M
Transport identifier	M

Result

This parameter provides information for the receiving application.

6.3.4.12.3 Service procedure

Since they are dependant on the transport class selected, service procedures for this service are detailed in IEC 61158-6-2.

6.4 Summary of FAL classes

Table 74 contains a summary of the defined FAL Classes.

Table 74 – FAL class summary

FAL ASE	Class	Class ID
OM ASE	General	na
	Identity	1
	Assembly	4
	Message Router	2
	Parameter	15
	Acknowledge Handler	43
	Time Sync	67
CM ASE	Connection Manager	6
CN ASE	Connection	5
AR ASE	UCMM – Client	na
	UCMM – Server	na
	Generic Transport	na
	Transport Class 0 – Client	na
	Transport Class 0 – Server	na
	Transport Class 1 – Client	na
	Transport Class 1 – Server	na
	Transport Class 2 – Client	na
	Transport Class 2 – Server	na
	Transport Class 3 – Client	na
	Transport Class 3 – Server	na

6.5 Permitted FAL services by AR type

Table 75 below defines the valid combinations of services and AR types (which service APDUs can be can sent or received by AR with the specified type). The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 75 – FAL services by AR type

	Unc	Cnf	Client / Sender	Server / Receiver	UCMM	TR0 – TR6
FAL Services			**req rcv**	**req rcv**		
OM ASE						
Get_Atribute_Single		X	X	X	X	TR3
Get_Attribute_List		X	X	X	X	TR3
Get_Attributes_All		X	X	X	X	TR3
Set_Attribute_Single		X	X	X	X	TR3
Set_Attribute_List		X	X	X	X	TR3
Set_Attributes_All		X	X	X	X	TR3
Create		X	X	X	X	TR3
Delete		X	X	X	X	TR3
Start		X	X	X	X	TR3
Stop		X	X	X	X	TR3
Reset		X	X	X	X	TR3
Find_Next_Instance		X	X	X	X	TR3
NOP		X	X	X	X	TR3
Apply_Attributes		X	X	X	X	TR3
Save		X	X	X	X	TR3
Restore		X	X	X	X	TR3
Get_Member		X	X	X	X	TR3
Set_Member		X	X	X	X	TR3
Insert_Member		X	X	X	X	TR3
Remove_Member		X	X	X	X	TR3
Group_Sync		X	X	X	X	TR3
Add_AckData_Path		X	X	X	X	TR3
Remove_AckData_Path		X	X	X	X	TR3
Get_Enum_String		X	X	X	X	TR3
Symbolic_Translation		X	X	X	X	TR3
Flash_LEDs		X	X	X	X	TR3
Multiple_Service_Packet		X	X	X	X	TR3
CM ASE						
CM_Open		X	X	X	X	
CM_Close		X	X	X	X	
CM_Unconnected_Send		X	X	X	X	
CM_Get_Connection_Data		X	X	X	X	
CM_Search_Connection_Data		X	X	X	X	
CM_Get_Connection_Owner		X	X	X	X	
CN ASE						
Connection_Bind		X	X	X	X	TR3
Producing_Application_Lookup		X	X	X	X	TR3
AR ASE						
UCMM_Create		X	X			
UCMM_Delete	X		X			
UCMM_Write		X	X	X	X	
UCMM_Abort	X		X	X		
TR_Write	X		X	X		TR0 – TR6
TR_Trigger	X		X			TR0 – TR3
TR_Verify	X		X	X		TR3, TR6
TR_Packet_Arrived	X			X		TR0 – TR6
TR_Ack_Received	X		X			TR2
TR_Status_Update	X		X	X		TR4, TR5

Bibliography

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 bibliographic references.

IEC 61131-1, *Programmable controllers – Part 1: General information*

IEC 61158-2:2014, *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61588:2004[12], *Precision clock synchronization protocol for networked measurement and control systems*

IEC 61784-1:2019, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61784-2:2019, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC/IEEE 8802-3*

IEC 62026-3, *Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet*

ISO/IEC/IEEE 8802-3, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Standard for Ethernet*

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*

ODVA: THE CIP NETWORKS LIBRARY – *Volume 1: Common Industrial Protocol (CIPTM) – Edition 3.22, April 2017*, available at <http://www.odva.org> [viewed 2018-09-04]

ODVA: THE CIP NETWORKS LIBRARY – *Volume 2: EtherNet/IPTM Adaptation of CIP – Edition 1.23, April 2017*, available at <http://www.odva.org> [viewed 2018-09-04]

ODVA: THE CIP NETWORKS LIBRARY – *Volume 3: DeviceNetTM Adaptation of CIP – Edition 1.14, November 2013*, available at <http://www.odva.org> [viewed 2018-09-04]

ODVA: THE CIP NETWORKS LIBRARY – *Volume 4: ControlNetTM Adaptation of CIP – Edition 1.8, April 2013,* available at <http://www.odva.org> [viewed 2018-09-04]

ODVA: THE CIP NETWORKS LIBRARY – *Volume 5: CIP SafetyTM – Edition 2.6, November 2012*, available at <http://www.odva.org> [viewed 2018-09-04]

12 This earlier edition is mentioned here and in the text for historical and comparison purposes.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-2: Définition des services de la couche application – Eléments de type 2

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC 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, l'IEC – 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 l'IEC"). 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 l'IEC, participent également aux travaux. L'IEC 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 l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC 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 l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC 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 l'IEC. L'IEC 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 l'IEC, 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 l'IEC, 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 l'IEC ou de toute autre Publication de l'IEC, 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 l'IEC peuvent faire l'objet de droits de brevet. L'IEC 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 protocoles sont spécifiées dans l'IEC 61784-1 et l'IEC 61784-2.

La Norme internationale IEC 61158-5-2 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels.

Cette quatrième édition annule et remplace la troisième édition parue en 2014. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

- ajout d'un type de données en 5.3.2;
- clarifications de l'ASE Object management en 6.2.1;
- extensions de l'ASE General en 6.2.1.2.1;
- extensions/clarifications de l'ASE Identity en 6.2.1.2.2;
- mise à jour de l'ASE Message Router en 6.2.1.2.4;
- extensions/clarifications de l'ASE Time Sync en 6.2.1.2.6;
- mises à jour de l'ASE Parameter en 6.2.1.2.7;
- mises à jour de la spécification des services des ASE de FAL en 6.2.1.3;
- extensions/clarifications de l'ASE Connection Manager en 6.2.2;
- extensions/clarifications de l'ASE Connection en 6.2.3;
- extensions/clarifications du type d'application en 6.3.1.4.5.
- corrections rédactionnelles diverses.

La présente version bilingue (2021-02) correspond à la version anglaise monolingue publiée en 2019-04.

La version française de cette norme n'a pas été soumise au vote.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

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

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC 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.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

Le présent document s'inscrit dans une série créée pour faciliter l'interconnexion des composants de systèmes d'automation. Il renvoie aux autres normes de l'ensemble défini par le modèle de référence de bus de terrain "à trois couches" décrit dans l'IEC 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 présent document définit les caractéristiques du service d'application que les applications de bus de terrain et/ou la gestion de système peuvent exploiter.

Dans l'ensemble de normes relatives aux bus de terrain, le terme "service" désigne une capacité abstraite fournie par une couche du modèle de référence de base de l'interconnexion des systèmes ouverts (Open Systems Interconnection, OSI) à la couche immédiatement supérieure. Ainsi, le service de couche application défini dans le présent document est un service architectural conceptuel, indépendant des divisions administratives et de mise en œuvre.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-2: Définition des services de la couche application – Eléments de type 2

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 partie de l'IEC 61158 fournit des éléments communs pour les communications à 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 2. Le terme "à temps critique" signale l'existence d'une fenêtre temporelle dans laquelle une ou plusieurs actions spécifiées doivent être réalisées, avec un niveau de certitude défini. La non-réalisation des actions spécifiées dans la fenêtre temporelle induit un risque de défaillance des applications qui demandent ces actions, avec les risques afférents pour l'équipement, les installations et éventuellement la vie humaine.

La présente Norme internationale définit de manière abstraite le service visible de l'extérieur fourni par la couche application de bus de terrain de type 2 en termes:

a) de modèle abstrait visant à la définition des ressources d'application (objets) pouvant être manipulées par des utilisateurs utilisant un service FAL;

b) d'événements et d'actions liés aux primitives du service;

c) de paramètres associés à chaque événement et action de primitive, ainsi que de forme prise par ces paramètres; et

d) d'interaction entre ces événements et ces actions, ainsi que de séquences valides desdits événements et actions.

Le présent document vise à définir les services mis en place pour:

a) l'utilisateur de FAL, à la frontière entre l'utilisateur et la couche application du modèle de référence de bus de terrain; et

b) la Gestion des systèmes, à la frontière entre la couche application et la Gestion des systèmes selon le modèle de référence de bus de terrain.

Le présent document spécifie la structure et les services de la couche application de bus de terrain de type 2, en conformité avec le modèle de référence de base de l'OSI (ISO/IEC 7498-1) et la structure de la couche application de l'OSI (ISO/IEC 9545).

Les services et protocoles de couche FAL sont fournis par des entités AE de couche FAL contenues dans les processus d'application. L'AE de la FAL se compose d'un jeu d'éléments de service application (Application Service Element, ASE) orientés objet et d'une entité de gestion de couche (Layer Management Entity, LME) qui gère l'AE. Les éléments ASE délivrent des services de communication agissant sur un ensemble de classes d'objets de processus d'application (Application Process Object, APO) associées. L'un des éléments ASE de couche FAL est un élément ASE de gestion qui fournit un ensemble commun de services destinés à la gestion des instances des classes de couche FAL.

Quoique ces services spécifient, du point de vue des applications, les modalités d'émission et de remise des demandes et des réponses, ils ne comprennent pas de spécification du traitement que doivent en faire les applications demandeuse et répondeuse. En d'autres termes, les aspects comportementaux des applications ne sont pas définis; seule une définition des demandes et réponses que ces applications peuvent envoyer/recevoir est établie. Cela laisse une plus grande marge de manœuvre aux utilisateurs de la couche FAL dans la normalisation du comportement de ces objets. Outre ces services, le présent document définit également certains services de soutien donnant accès à la couche FAL dans un but de commande de certains aspects de son fonctionnement.

1.2 Spécifications

Le présent document a pour principal objet de préciser les caractéristiques des services conceptuels de couche application adaptés aux communications à temps critique; elle vise ainsi à compléter le modèle de référence de base OSI en guidant le développement de protocoles de couche application destinés aux communications à temps critique.

Un objectif secondaire consiste à fournir des voies d'évolution à partir des protocoles de communication industriels antérieurs. Ce dernier objectif explique la diversité des services normalisés sous la forme des différents Types IEC 61158, ainsi que celle des protocoles correspondants, normalisés dans les sous-parties de l'IEC 61158-6.

La présente spécification peut être utilisée comme la base pour les interfaces de programmation d'applications (Application Programming-Interfaces) formelles. Cependant, elle ne constitue pas une interface de programmation formelle, et toute interface de ce type devra faire face à des problèmes de mise en œuvre non couverts par la présente spécification, notamment

a) les dimensions et l'ordre des octets de plusieurs paramètres de service multioctet, et

b) la corrélation des primitives associées (demande et confirmation, ou indication et réponse).

1.3 Conformité

Le présent document ne définit pas de mises en œuvre ni de produits particuliers, pas plus qu'il ne limite les mises en œuvre des entités de couche application dans les systèmes d'automation industriels.

Il n'existe pas de conformité de l'équipement à la présente norme de définition de service de couche Application. Au contraire, la conformité est obtenue par une mise en œuvre de protocoles conformes de couche application qui satisfont aux services de couche application de type 2 définis dans le présent document.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. 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 IEC 61158, ainsi que de l'IEC 61784-1 et de l'IEC 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.

IEC 61131-3:2003[1], *Automates programmables – Partie 3: Langages de programmation*

[1] Une édition plus récente de cette norme a été publiée, mais seule l'édition citée s'applique.

IEC 61158-1:2019, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Vue d'ensemble et recommandations pour les séries IEC 61158 et IEC 61784*

IEC 61158-3-2:2014, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-2: Définition des services de la couche liaison de données – Eléments de type 2*

IEC 61158-3-2:2014/AMD1:2019

IEC 61158-4-2:2019, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-2: Spécification du protocole de la couche liaison de données – Eléments de type 2*

IEC 61158-6-2:2019, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-2: Spécification du protocole de la couche application – Eléments de type 2*

IEC 61588:2009, *Precision clock synchronization protocol for networked measurement and control systems* (disponible en anglais seulement)

IEC 61784-3-2, *Réseaux de communication industriels – Profils – Partie 3-2: Bus de terrain de sécurité fonctionnelle – Spécifications supplémentaires pour CPF 2*

ISO/IEC 646, *Technologies de l'information – Jeu ISO de caractères codés à 7 éléments pour l'échange d'information*

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

ISO/IEC 8859-1, *Technologies de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 1 Alphabet latin n° 1*

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

ISO/IEC 10646, *Technologies de l'information – Jeu universel de caractères codés (JUC)*

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

ISO/IEC/IEEE 60559, *Technologies de l'information – Systèmes de microprocesseurs – Arithmétique flottante*

ISO 639-2, *Codes pour la représentation des noms de langue – Partie 2: Code alpha-3*

ISO 8859-1[2]:1987, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 1: Alphabet latin n° 1*

ISO 8859-2[3]:1987, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 2: Alphabet latin n° 2*

---

2 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

3 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

ISO 8859-3[4]:1988, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 3: Alphabet latin nº 3*

ISO 8859-4[5]:1988, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 4: Alphabet latin nº 4*

ISO 8859-5[6]:1988, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 5: Alphabet latin/cyrillique*

ISO 8859-6[7]:1987, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 6: Alphabet latin/arabe*

ISO 8859-7[8]:1987, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 7: Alphabet latin/grec*

ISO 8859-8[9]:1988, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 8: Alphabet latin/hébreu*

ISO 8859-9[10]:1989, *Traitement de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 9: Alphabet latin nº 5*

ISO 11898:1993[11], *Véhicules routiers – Echange d'information numérique – Gestionnaire de réseau de communication à vitesse élevée (CAN)*

IETF RFC 1759, *Printer MIB,* disponible à l'adresse <http://www.ietf.org> [consultée le 04/09/2018]

3 Termes, définitions, symboles, termes abrégés et conventions

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

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

---

4 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

5 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

6 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

7 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

8 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

9 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

10 Une version plus récente de cette norme a été publiée par ISO/IEC, mais l'édition citée s'applique aux normes IETF référencées.

11 Une édition plus récente de cette norme a été publiée, mais seule l'édition citée s'applique.

3.1 Termes de l'ISO/IEC 7498-1

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

3.2 Termes de l'ISO/IEC 8822

a) syntaxe abstraite
b) contexte de présentation

3.3 Termes de l'ISO/IEC 9545

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

3.4 Termes de l'ISO/IEC 8824-1

a) identificateur d'objet
b) type

3.5 Termes pour la couche Liaison de données de bus de terrain de type 2

Les termes suivants, définis dans l'IEC 61158-3-2 et l'IEC 61158-4-2, 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) étiquette fixe
j) étiquette générique
k) liaison
l) ID MAC
m) adresse réseau

n) adresse de nœud

o) nœud

p) étiquette

q) programmé

r) non programmé

3.6 Définitions relatives à la couche application de bus de terrain de type 2

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

3.6.1
allouer
prendre une ressource dans une zone commune et affecter la ressource en question à l'usage exclusif d'une entité spécifique

3.6.2
application
fonction ou structure de données pour laquelle des données sont consommées ou produites

3.6.3
objets d'application
classes d'objets multiples qui gèrent et assurent un échange de messages pendant le mode exécution à travers le réseau et à l'intérieur de l'appareil de réseau

3.6.4
processus d'application
partie d'une application distribuée sur un réseau, qui est située sur un appareil et adressée sans ambiguïté

3.6.5
objet de processus d'application
composant d'un processus d'application, identifiable et accessible via une relation d'applications de la FAL

3.6.6
classe d'objets de processus d'application
classe d'objets de processus d'application définis par un ensemble de services et d'attributs accessibles via le réseau

3.6.7
relation d'applications
association de type coopératif entre deux invocations d'entités d'application (application-entity-invocation) ou plus, dans un but d'échange d'informations et de coordination de leur action conjointe

Note 1 à l'article: Cette relation est activée soit par l'échange d'unités de données de protocole d'application (application-protocol-data-unit), soit à la suite d'activités de préconfiguration.

3.6.8
élément de service d'application de relation d'applications
élément de service d'application (application-service-element) qui fournit le seul moyen d'établir et de rompre toute relation d'applications

3.6.9
point d'extrémité de relation d'applications
contexte et comportement d'une relation d'applications, vus et maintenus par un des processus d'application impliqués dans la relation d'applications

Note 1 à l'article: Chaque processus d'application impliqué dans la relation d'applications maintient son propre point d'extrémité de relation d'applications.

3.6.10
attribut
description d'une caractéristique, visible par un observateur externe, d'un objet

Note 1 à l'article: Les attributs d'un objet contiennent des informations sur les portions variables d'un objet. En règle générale, ils fournissent des informations de statut ou régissent l'action d'un objet. Les attributs peuvent également influer sur le comportement d'un objet. Les attributs se répartissent en deux catégories: les attributs de classe et les attributs d'instance.

3.6.11
comportement
indication de la manière dont un objet réagit à des événements particuliers

3.6.12
Algorithme de la meilleure "horloge mère"
BMCA
algorithme exécuté par chaque nœud afin d'identifier l'horloge qui deviendra l'horloge mère d'un sous-réseau et l'horloge grand-mère du domaine

Note 1 à l'article: L'algorithme compare principalement l'attribut priority1, la qualité d'horloge, l'attribut priority2 et l'identité de la source pour identifier la meilleure mère parmi les différentes candidates.

3.6.13
horloge frontière
horloge possédant plusieurs ports PTP (Precision Time Protocol) dans un même domaine, qui maintient l'échelle de temps utilisée dans ce domaine

Note 1 à l'article: Une horloge frontière peut jouer le rôle de source de temps, c'est-à-dire être une horloge mère, ou se synchroniser à une autre horloge, c'est-à-dire être une horloge esclave.

[SOURCE: IEC 61588:2009, 3.1.3, modifiée – deuxième phrase transformée en Note]

3.6.14
classe
ensemble d'objets représentant le même type de composant du système

Note 1 à l'article: Une classe est une généralisation d'un objet, un modèle qui permet de définir des variables et des méthodes. Tous les objets d'une classe possèdent une forme et un comportement identiques, mais leurs attributs contiennent généralement des données différentes.

3.6.15
attribut de classe
attribut commun à tous les objets d'une classe

3.6.16
code de classe
identificateur unique attribué à chaque classe d'objets

3.6.17
service spécifique à une classe
service défini par une classe d'objets particulière afin de remplir une fonction nécessaire qui n'est assurée par aucun service commun

Note 1 à l'article: Tout objet spécifique à une classe est réservé à l'usage exclusif de la classe d'objets qui le définit.

3.6.18
client
a) objet qui utilise les services d'un autre objet (serveur) pour réaliser une tâche

b) initiateur d'un message auquel un serveur réagit

3.6.19
horloge
nœud participant au protocole PTP, capable de donner une mesure de l'écoulement du temps depuis une époque définie

Note 1 à l'article: Il y a trois types d'horloges dans l'IEC 61588:2009: les horloges frontières, les horloges transparentes et les horloges ordinaires.

[SOURCE: IEC 61588:2009, 3.1.4, modifiée – Note différente]

3.6.20
objets de communication
composants qui gèrent et assurent un échange de messages pendant le mode exécution à travers le réseau

EXEMPLES objet Connection Manager (gestionnaire de connexion), objet Unconnected Message Manager (UCMM, gestionnaire de messages non connectés), et objet Message Router (routeur de message).

3.6.21
connexion
liaison logique entre deux objets d'application qui peuvent se trouver au sein du même appareil ou dans des appareils différents

Note 1 à l'article: Les connexions peuvent être soit point à point, soit multipoint.

3.6.22
ID de connexion
CID
identificateur affecté à une émission qui est associé à une connexion particulière entre producteurs et consommateurs, donnant un nom à un élément spécifique d'information d'application

3.6.23
chemin de connexion
train d'octets qui définit l'objet d'application auquel une instance de connexion s'applique

3.6.24
point de connexion
tampon qui est représenté comme étant une sous-instance d'un objet Assembly (ensemble)

3.6.25
consommer
acte consistant à recevoir des données provenant d'un producteur

3.6.26
consommateur
nœud ou puits qui reçoit des données d'un producteur

3.6.27
application consommatrice
application qui consomme des données

3.6.28
cyclique
qui se reproduit de manière régulière et répétitive

3.6.29
appareil
matériel physique connecté à la liaison

Note 1 à l'article: Un appareil peut contenir plusieurs nœuds.

3.6.30
profil d'appareil
ensemble d'informations et de fonctionnalités dépendantes de l'appareil qui garantit la cohérence entre les appareils similaires appartenant au même type d'appareil

3.6.31
domaine
groupement logique d'horloges synchronisées entre elles à l'aide du protocole, mais pas nécessairement synchronisées aux horloges d'un autre domaine

[SOURCE: IEC 61588:2009, 3.1.7]

3.6.32
nœud d'extrémité
nœud producteur ou consommateur

3.6.33
point d'extrémité
une des entités en communication impliquées dans une connexion

3.6.34
époque
origine d'une échelle de temps

[SOURCE: IEC 61588:2009, 3.1.9]

3.6.35
erreur
divergence entre une valeur ou condition calculée, observée ou mesurée et la valeur ou condition spécifiée ou théoriquement correcte

3.6.36
événement
instance d'un changement de conditions

3.6.37
trame
synonyme critiqué de DLPDU

3.6.38
horloge grand-mère
dans un domaine, horloge constituant la source de temps primaire pour la synchronisation d'horloge à l'aide du protocole PTP

[SOURCE: IEC 61588:2009, 3.1.13]

3.6.39
groupe
a) <généralités> terme général pour un ensemble d'objets. Usages spécifiques:

b) <adressage> lors de la description d'une adresse, adresse qui identifie plus d'une entité

3.6.40
interface
(a) frontière commune entre deux unités fonctionnelles, définie par des caractéristiques fonctionnelles, des caractéristiques de signal ou d'autres caractéristiques adaptées

(b) ensemble d'attributs et de services de classe FAL représentant une vue spécifique de la classe FAL

3.6.41
invocation
acte d'utiliser un service ou une autre ressource d'un processus d'application

Note 1 à l'article: Chaque invocation représente un fil de commande distinct qui peut être décrit par son contexte. Une fois le service terminé ou la ressource libérée, l'invocation cesse d'exister. Dans le cas des invocations de services, un service lancé, mais pas encore terminé est considéré comme une invocation de services en cours. Dans le cas des invocations de services, un ID d'invocation (Invoke ID) peut être utilisé pour identifier une invocation de services de manière non ambiguë et la différencier des autres invocations de services en cours.

3.6.42
instance
occurrence physique permettant d'identifier un objet parmi d'autres au sein d'une même classe d'objets

EXEMPLE "California" (La Californie) est une instance de la classe d'objets "state" (état).

Note 1 à l'article: Les termes objet, instance et instance d'objet font référence à une instance particulière.

3.6.43
attribut d'instance
attribut dont la valeur est propre à une instance d'objet et dont la définition est partagée par toutes les instances d'un objet

3.6.44
instancié
se dit d'un objet créé dans un appareil

3.6.45
appareil logique
classe FAL particulière qui extrait un composant logiciel ou de microprogramme sous la forme d'un dispositif intégré et autonome au sein d'un appareil d'automation

3.6.46
Lpacket
Link packet
élément d'information d'application qui contient une taille, un octet de contrôle, une étiquette et des données de liaison

Note 1 à l'article: Les couches Liaison de données des homologues utilisent des Lpacket pour envoyer et recevoir des unités de données de service provenant des couches supérieures de la pile OSI.

3.6.47
informations de gestion
informations accessibles via le réseau qui facilitent la gestion de l'exploitation du système de bus de terrain, y compris la couche application

Note 1 à l'article: La gestion inclut des fonctions telles que la commande, la surveillance et le diagnostic.

3.6.48
horloge mère
dans le cas d'un chemin de communication PTP unique, horloge constituant la source de temps à laquelle toutes les autres horloges de ce chemin se synchronisent

[SOURCE: IEC 61588:2009, 3.1.17]

3.6.49
membre
élément d'un attribut qui est structuré comme une partie d'une matrice

3.6.50
Message Router
objet au sein d'un nœud qui distribue les demandes de messagerie vers les objets d'application appropriés

3.6.51
méthode
<objet> synonyme d'un service opérationnel délivré par l'élément ASE serveur et invoqué par un client

3.6.52
module
composant matériel ou logique d'un appareil physique

3.6.53
connexion multipoint
connexion d'un nœud à plusieurs autres nœuds

Note 1 à l'article: Les connexions multipoint permettent que les messages issus d'un seul producteur soient reçus par plusieurs nœuds consommateurs.

3.6.54
réseau
ensemble de nœuds reliés par un support de communication d'un type ou d'un autre, avec d'éventuels répétiteurs intermédiaires, ponts, routeurs et passerelles de couche inférieure

3.6.55
objet
représentation abstraite d'un composant particulier dans un appareil, généralement un ensemble de données (sous la forme de variables) et de méthodes (procédures) associées, destiné à l'exploitation des données dont l'interface et le comportement sont clairement définis

3.6.56
service spécifique à un objet
service réservé à l'usage exclusif de la classe d'objets qui le définit

3.6.57
horloge ordinaire
horloge possédant un seul port PTP dans un même domaine, qui maintient l'échelle de temps utilisée dans ce domaine

Note 1 à l'article: Une horloge ordinaire peut jouer le rôle de source de temps, c'est-à-dire être une horloge mère, ou se synchroniser à une autre horloge, c'est-à-dire être une horloge esclave.

[SOURCE: IEC 61588:2009, 3.1.22, modifiée – deuxième phrase transformée en Note]

3.6.58
émetteur
client chargé d'établir un chemin de connexion vers une cible

3.6.59
horloge parente
horloge mère à laquelle une horloge est synchronisée

[SOURCE: IEC 61588:2009, 3.1.23]

3.6.60
homologue
rôle d'un point d'extrémité d'AR dans lequel il est capable d'agir à la fois comme client et comme serveur

3.6.61
appareil physique
appareil d'automation ou autre appareil de réseau

3.6.62
connexion point à point
connexion établie précisément entre deux objets d'application

3.6.63
Precision Time Protocol
PTP
protocole défini par l'IEC 61588:2009

Note 1 à l'article: Utilisé comme adjectif, ce terme indique que le nom modifié est spécifié ou interprété dans le contexte de l'IEC 61588:2009.

[SOURCE: IEC 61588:2009, 3.1.28, modifiée – deuxième phrase transformée en Note]

3.6.64
produire
acte d'envoyer des données destinées à être reçues par un consommateur

3.6.65
producteur
nœud chargé de l'envoi des données

3.6.66
propriété
terme général désignant toute information descriptive concernant un objet

3.6.67
message PTP
un des types de messages définis dans l'IEC 61588:2009

[SOURCE: IEC 61588:2009, 3.1.33]

3.6.68
port PTP
point d'accès logique d'une horloge pour les communications PTP, au réseau de communication

[SOURCE: IEC 61588:2009, 3.1.35]

3.6.69
ressource
capacité de traitement ou d'information d'un sous-système

3.6.70
numéro de série
valeur entière unique de 32 bits attribuée par chaque fabricant/vendeur à chaque appareil ayant des capacités de communication de type 2

Note 1 à l'article: L'identificateur de vendeur et le numéro de série forment conjointement un identificateur propre à chaque appareil.

3.6.71
serveur
a) rôle d'un AREP dans lequel il renvoie une unité APDU de réponse de service confirmé au client à l'origine de la demande

b) objet qui fournit des services à un autre objet (client)

3.6.72
service
opération ou fonction qu'un objet et/ou une classe d'objets réalise ou assure à la demande d'un autre objet et/ou d'une autre classe d'objets

3.6.73
horloges synchronisées
(avec une incertitude spécifiée) deux horloges qui ont la même époque et pour lesquelles les mesures de la durée d'un événement unique à une heure arbitraire ne diffèrent pas de plus que l'incertitude spécifiée

[SOURCE: IEC 61588:2009, 3.1.40, modifiée – reformulée]

3.6.74
heure système
temps absolu tel que défini par la synchronisation de CPF2 dans le contexte d'un système de temps distribué où tous les appareils ont une horloge locale qui est synchronisée avec une horloge mère commune

Note 1 à l'article: Dans le contexte de CPF2, System Time (heure système) est une valeur entière de 64 bits exprimée en unités de nanosecondes avec une valeur 0 correspondant à la date 1970-01-01 (c'est-à-dire 1er janvier 1970).

3.6.75
cible
nœud d'extrémité auquel une connexion est établie

3.6.76
horloge transparente
appareil qui mesure le temps nécessaire à un message d'événement PTP pour traverser l'appareil et qui fournit cette information aux horloges recevant ce message d'événement PTP

[SOURCE: IEC 61588:2009, 3.1.46, modifiée –tronqué]

3.6.77
gestionnaire de messages non connectés
UCMM
composant au sein d'un nœud qui émet et reçoit des messages explicites non connectés, et les envoie directement à l'objet Message Router (routeur de message)

3.6.78
service non connecté
service de messagerie qui ne repose pas sur l'établissement d'une connexion entre les appareils avant d'autoriser des échanges d'informations

3.6.79
identificateur de vendeur
identification de chaque fabricant/vendeur de produit par un numéro unique

Note 1 à l'article: Les identificateurs de vendeur sont attribués par l'ODVA, Inc.

3.7 Termes abrégés et symboles de type 2

AE Application Entity (Entité d'application)
AL Application Layer (Couche application)
APO Application Object (Objet d'application)
AP Application Process (Processus d'application)
APDU Application Protocol Data Unit (Unité de données de protocole d'application)
AR Application Relationship (Relation d'applications)
AREP Application Relationship End Point (Point d'extrémité de relation d'applications)
ASCII American Standard Code for Information Interchange (Code américain normalisé pour l'échange d'informations)
ASE Application service element (Elément de service d'application)
CID Connection ID (ID de connexion)
CM_API Actual Packet Interval (Intervalle réel entre paquets)
CM_RPI Requested Packet Interval (Intervalle demandé entre paquets)
Cnf Confirmation (Confirmation)
CR Communication Relationship (Relation de communication)
DL- (as a prefix) data Link ((comme préfixe) liaison de données)
DLC Data-Link Connection (Connexion de liaison de données)
DLCEP Data-Link Connection End Point (Point d'extrémité de connexion de liaison de données)
DLL Data-Link Layer (Couche Liaison de données)
DLM Data-Link Management (Gestion de liaison de données)
DLSAP Data-Link Service Access Point (Point d'accès au service de liaison de données)
DLSDU DL-Service-Data-Unit (Unité de données de service de liaison de données)
FAL Fieldbus Application Layer (Couche application de bus de terrain)
FIFO First-In First-Out (Premier entré premier sorti)

ID Identifier (Identificateur)

IEC International Electrotechnical Commission (Commission Électrotechnique Internationale)

Ind Indication (Indication)

IP Internet Protocol (Protocole Internet)

ISO International Organization for Standardization (Organisation internationale de normalization)

I/O Input/output (Entrée/Sortie)

LME Layer Management Entity (Entité de gestion de couche)

O2T Originator to Target (Caractéristiques de connexion (émetteur vers cible))

O⇒T Originator to Target (Caractéristiques de connexion(émetteur vers cible))

OSI Open Systems Interconnect (Interconnexion de systèmes ouverts)

PDU Protocol Data Unit (Unité de données de protocole)

PL Physical Layer (Couche physique)

PTP Precision Time Protocol) (Protocole de précision temporelle) [IEC 61588:2009]

QoS Quality of service (Qualité de service)

REP Route End Point (Point d'extrémité de chemin)

Req Request (Demande)

Rsp Response (Réponse)

SAP Point d'accès au service (Service Access Point)

SDU Service-Data-Unit (Unité de données de service)

SEM State Event Matrix (Matrice d'événements d'états

STD State Transition Diagram (Diagramme de transitions d'états, utilisé pour décrire le comportement d'objet))

T2O Target to Originator (Cible vers émetteur (caractéristiques de connexion))

T⇒O Target to Originator (Cible vers émetteur (caractéristiques de connexion))

3.8 Conventions

3.8.1 Vue d'ensemble

La couche FAL se compose d'un ensemble d'éléments ASE orientés objet. Chaque élément ASE est défini dans un paragraphe distinct. Chaque spécification d'ASE est constituée de deux parties, à savoir sa spécification de classe et sa spécification de service.

La spécification de classe définit les attributs de la classe. Les attributs sont accessibles à partir d'instances de la classe en utilisant les services d'ASE de gestion d'objets spécifiés à l'Article 5 du présent document. La spécification de service définit les services fournis par l'élément ASE.

3.8.2 Conventions générales

Le présent document emploie les conventions de description énoncées dans l'ISO/IEC 10731.

La police gras est utilisée dans le présent document pour mettre en valeur des noms de paramètre ou des éléments importants dans le texte.

3.8.3 Conventions pour les définitions de classe

Les définitions de classe sont décrites à l'aide de modèles. Chaque modèle se compose d'une liste d'attributs associés à la classe. La forme générale du modèle est montrée ci-dessous:

FAL ASE:			**nom de l'ASE**
CLASS:			**Nom de la classe**
CLASS ID:			**#**
PARENT CLASS:			nom de la classe parent
ATTRIBUTES:			
1	(o)	Key Attribute:	identifiant numérique
2	(o)	Key Attribute:	nom
3	(m)	Attribute:	nom d'attribut(valeurs)
4	(m)	Attribute:	nom d'attribut(valeurs)
4.1	(s)	Attribute:	nom d'attribut(valeurs)
4.2	(s)	Attribute:	nom d'attribut(valeurs)
4.3	(s)	Attribute:	nom d'attribut(valeurs)
5.	(c)	Constraint:	expression de la contrainte
5.1	(m)	Attribute:	nom d'attribut(valeurs)
5.2	(o)	Attribute:	nom d'attribut(valeurs)
6	(m)	Attribute:	nom d'attribut(valeurs)
6.1	(s)	Attribute:	nom d'attribut(valeurs)
6.2	(s)	Attribute:	nom d'attribut(valeurs)
SERVICES:			
1	(o)	OpsService:	nom de service
2.	(c)	Constraint:	expression de la contrainte
2.1	(o)	OpsService:	nom de service
3	(m)	MgtService:	nom de service

(1) La rubrique "FAL ASE:" est le nom de l'élément ASE de la couche FAL (FAL ASE) qui fournit les services pour la classe spécifiée.

(2) La rubrique "CLASS:" est le nom de la classe spécifiée. Tous les objets définis à l'aide de ce modèle seront une instance de cette classe. La classe peut être spécifiée par le présent document ou par un utilisateur du présent document.

(3) La rubrique "CLASS ID:" est un numéro qui identifie la classe spécifiée. Ce numéro est unique au sein du FAL ASE qui fournira les services pour cette classe. Lorsqu'il est qualifié par l'identité de son FAL ASE, il identifie sans ambiguïté la classe relevant du domaine d'application de la FAL. La valeur "NULL" indique que la classe ne peut pas être instanciée. Les Class ID (identificateurs de classe) entre 1 et 99, 240 et 767 sont réservés par le présent document pour identifier des classes normalisées. Les CLASS ID entre 100 et 199, 768 et 1 279 sont alloués pour identifier les classes définies par l'utilisateur.

(4) La rubrique "PARENT CLASS:" est le nom de la classe parent pour la classe spécifiée. Tous les attributs définis pour la classe parent et hérités par celle-ci sont hérités pour la classe définie, et ils n'ont donc pas à être redéfinis dans le modèle pour cette classe.

NOTE La classe parent "TOP" indique que la classe définie est une définition de classe initiale. La classe parent "TOP" est utilisée comme point de départ à partir duquel toutes les autres classes sont définies. L'usage de "TOP" est réservé pour les classes définies par le présent document.

(5) L'étiquette "ATTRIBUTES" (ATTRIBUTS) indique que les entrées suivantes sont des attributs définis pour la classe.

a) Chacune des entrées d'attribut contient un numéro de ligne dans la colonne 1, un indicateur obligatoire (m) / facultatif (o) / conditionnel (c) / sélecteur (s) dans la colonne 2, une étiquette de type d'attribut dans la colonne 3, un nom ou une expression conditionnelle dans la colonne 4, et, facultativement, une liste de valeurs énumérées dans la colonne 5. Dans la colonne suivant la liste de valeurs, la valeur par défaut pour l'attribut peut être spécifiée.

Si un attribut est facultatif, sa valeur par défaut (spécifiée dans l'IEC 61158-6-2) doit entraîner le même comportement que si l'attribut n'avait pas été implémenté.

b) Les objets sont normalement identifiés par un identificateur numérique et/ou par un nom d'objet. Dans les modèles de classe, ces attributs clés sont définis sous l'attribut clé.

c) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par un point (period). L'imbrication est utilisée pour spécifier

   i) des champs d'un attribut structuré (4.1, 4.2, 4.3),

   ii) des attributs conditionnés à un énoncé de contrainte (5). Les attributs peuvent être obligatoires (5.1) ou facultatifs (5.2) si la contrainte est vraie. Tous les attributs facultatifs n'exigent pas des énoncés de contraintes comme le fait l'attribut défini en 5.2,

   iii) les champs sélection d'un attribut de type choix (6.1 et 6.2).

(6) L'étiquette "SERVICES" indique que les entrées suivantes sont des services définis pour la classe.

a) Un (m) dans la colonne 2 indique que le service est obligatoire pour la classe, alors qu'un (o) indique qu'il est facultatif. Un (c) dans cette colonne indique que le service est conditionnel. Lorsque tous les services définis pour une classe le sont comme étant facultatifs, l'un au moins doit être sélectionné quand une instance de la classe est définie.

b) L'étiquette "OpsService" désigne un service opérationnel (1).

c) L'étiquette "MgtService" désigne un service de gestion (2).

d) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par un point (period). L'imbrication dans la liste de services sert à spécifier des services conditionnés à un énoncé de contrainte.

3.8.4 Conventions pour les définitions de service

3.8.4.1 Généralités

Le modèle de service, les primitives de service et les diagrammes séquentiels de temps utilisés sont des descriptions entièrement abstraites; ils ne représentent pas une spécification de mise en œuvre.

3.8.4.2 Paramètres du service

Les primitives de service sont utilisées pour représenter les interactions entre utilisateur de service et fournisseur de service (ISO/IEC 10731). Elles acheminent des paramètres qui indiquent des informations disponibles dans l'interaction entre utilisateur et fournisseur. Pour toute interface particulière, il n'est pas nécessaire d'indiquer explicitement tous les paramètres.

Les spécifications de service selon le présent document utilisent un format de tableau pour décrire les paramètres de composants des primitives du service d'ASE. Les paramètres qui s'appliquent à chaque groupe de primitives de service sont consignés en tableaux. Chaque tableau comporte jusqu'à cinq colonnes pour le/la:

1) nom de paramètre,
2) primitive "request",
3) primitive "indication",
4) primitive "response", et
5) primitive "confirm".

Un paramètre (ou un composant de celui-ci) est énuméré dans chaque ligne de chaque tableau. Dans les colonnes appropriées de la primitive de service, un code est utilisé pour spécifier le type d'usage du paramètre sur la primitive spécifiée dans la colonne:

M le paramètre est obligatoire pour la primitive;

U le paramètre est une option de l'utilisateur et peut ou peut ne pas être fourni, en fonction de l'usage dynamique de l'utilisateur du service. Lorsque ce paramètre n'est pas fourni, une valeur par défaut est supposée;

C le paramètre est conditionné à d'autres paramètres ou à l'environnement de l'utilisateur du service;

— – (blanc/vide) le paramètre n'est jamais présent;

S le paramètre est un élément sélectionné.

Certaines entrées sont par ailleurs qualifiées par des éléments entre parenthèses. Ces entrées peuvent être:

a) une contrainte spécifique au paramètre:

"(=)" indique que le paramètre équivaut du point de vue de la sémantique au paramètre dans la primitive de service située immédiatement à sa gauche dans le tableau;

b) une indication qu'une certaine note s'applique à l'entrée:

"(n)" indique que la note "n" suivante contient des informations complémentaires relatives au paramètre et à son utilisation.

3.8.4.3 Procédures de service

Les procédures sont définies en termes des

- interactions entre entités d'application par l'échange d'unités de données de protocole d'application de bus de terrain, et
- interactions entre un fournisseur de service de couche application et un utilisateur de service de couche application dans le même système par l'invocation de primitives de service de couche application.

Ces procédures sont applicables à des instances de communication entre systèmes qui prennent en charge des services de communication à contrainte temporelle au sein de la couche application de bus de terrain.

4 Concepts communs

Les concepts et modèles communs utilisés pour décrire le service de couche application dans le présent document sont détaillés dans l'Article 9 de l'IEC 61158-1.

5 Elément ASE de type de données

5.1 Généralités

Une vue d'ensemble de l'ASE de type de données et des relations entre types de données est fournie dans le paragraphe 10.2 de l'IEC 61158-1.

5.2 Définition formelle des objets de type de données

Le modèle utilisé pour décrire la classe de types de données de l'Article 5 est détaillé dans le paragraphe 10.2 de l'IEC 61158-1. Il inclut la structure spécifique à l'élément ASE et la définition de ses attributs.

5.3 Types de données définis par la couche FAL

5.3.1 Types Fixed length (longueur fixe)

5.3.1.1 Types Boolean (booléens)

5.3.1.1.1 Boolean

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	1
2	Data type Name	=	Boolean
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

Ce type de données exprime un type de données Boolean avec les valeurs TRUE et FALSE.

5.3.1.1.2 BOOL

Ce type de l'IEC 61131-3 est le même que le type Boolean.

5.3.1.2 Types Bitstring (chaîne de bits)

5.3.1.2.1 BitString8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	22
2	Data type Name	=	BitString8
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	1

Ce type contient un élément de type BitString.

5.3.1.2.2 SWORD

Ce type de l'IEC 61131-3 est le même que le type Bitstring8.

5.3.1.2.3 BitString16

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	23
2	Data type Name	=	Bitstring16
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	2

5.3.1.2.4 WORD

Ce type de l'IEC 61131-3 est le même que le type Bitstring16.

5.3.1.2.5 BitString32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	24
2	Data type Name	=	Bitstring32
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	4

5.3.1.2.6 DWORD

Ce type de l'IEC 61131-3 est le même que le type Bitstring32.

5.3.1.2.7 BitString64

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	57
2	Data type Name	=	Bitstring64
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	8

5.3.1.2.8 LWORD

Ce type de l'IEC 61131-3 est le même que le type Bitstring64.

5.3.1.3 Types Date

5.3.1.3.1 DATE

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	DATE
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

Ce type de l'IEC 61131-3 est un nombre binaire. Le bit de poids le plus fort de l'octet de poids le plus fort est toujours utilisé comme bit de poids le plus fort du nombre binaire; aucun bit de signe n'est inclus. Ce type Unsigned (non signé) a une longueur de deux octets. Il exprime la date comme un nombre de jours, du 1972-01-01 (1er janvier 1972), début de l'ère du Temps Universel Coordonné (TUC), jusqu'au 2151-06-06 (6 juin 2151), c'est-à-dire une plage totale de 65 536 jours.

5.3.1.3.2 TimeOfDay

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

Ce type de données est constitué de deux éléments de valeurs unsigned (non signées) et exprime l'heure du jour et la date. Le premier élément est un type de données Unsigned32 et donne l'heure après minuit en millisecondes. Le second élément est un type de données Unsigned16 qui donne la date en comptant les jours à partir de 1984-01-01 (1er janvier 1984).

5.3.1.3.3 TimeOfDay without date indication (sans indication de date)

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	52
2	Data type Name	=	TimeOfDay without date indication
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Ce type de données est constitué d'un seul élément d'une valeur non signée et exprime l'heure du jour. L'élément est un type de données Unsigned32 et donne l'heure après minuit en millisecondes.

5.3.1.3.4 TIME_OF_DAY

Ce type de l'IEC 61131-3 est le même que le type TimeofDay sans indication de date.

5.3.1.4 Types numériques

5.3.1.4.1 Types Floating Point (virgule flottante)

5.3.1.4.1.1 Float32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	8
2	Data type Name	=	Float32
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Ce type a une longueur de quatre octets. Le format de Float32 est celui défini par l'ISO/IEC/IEEE 60559 simple précision.

5.3.1.4.1.2 REAL

Ce type de l'IEC 61131-3 est le même que le type Float32.

5.3.1.4.1.3 Float64

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	15
2	Data type Name	=	Float64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

Ce type a une longueur de huit octets. Le format de Float64 est celui défini par l'ISO/IEC/IEEE 60559 double précision.

5.3.1.4.1.4 LREAL

Ce type de l'IEC 61131-3 est le même que le type Float64.

5.3.1.4.2 Types Integer (entiers)

5.3.1.4.2.1 Integer8

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2
2	Data type Name	=	Integer8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

Ce type entier est un nombre binaire en complément à deux d'une longueur d'un octet.

5.3.1.4.2.2 SINT

Ce type de l'IEC 61131-3 est le même que le type Integer8.

5.3.1.4.2.3 Integer16

CLASS: **Data type**
ATTRIBUTES:

1	Data type Numeric Identifier	=	3
2	Data type Name	=	Integer16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

Ce type entier est un nombre binaire en complément à deux d'une longueur de deux octets.

5.3.1.4.2.4 INT

Ce type de l'IEC 61131-3 est le même que le type Integer16.

5.3.1.4.2.5 Integer32

CLASS: **Data type**
ATTRIBUTES:

1	Data type Numeric Identifier	=	4
2	Data type Name	=	Integer32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Ce type entier est un nombre binaire en complément à deux d'une longueur de quatre octets.

5.3.1.4.2.6 DINT

Ce type de l'IEC 61131-3 est le même que le type Integer32.

5.3.1.4.2.7 Integer64

CLASS: **Data type**
ATTRIBUTES:

1	Data type Numeric Identifier	=	55
2	Data type Name	=	Integer64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

Ce type entier est un nombre binaire en complément à deux d'une longueur de huit octets.

5.3.1.4.2.8 LINT

Ce type de l'IEC 61131-3 est le même que le type Integer64.

5.3.1.4.3 Types Unsigned (non signés)

5.3.1.4.3.1 Unsigned8

CLASS: **Data type**
ATTRIBUTES:

1	Data type Numeric Identifier	=	5
2	Data type Name	=	Unsigned8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

Ce type est un nombre binaire. Le bit de poids le plus fort de l'octet de poids le plus fort est toujours utilisé comme bit de poids le plus fort du nombre binaire; aucun bit de signe n'est inclus. Ce type a une longueur d'un octet.

5.3.1.4.3.2 USINT

Ce type de l'IEC 61131-3 est le même que le type Unsigned8.

5.3.1.4.3.3 Unsigned16

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	6
2	Data type Name	=	Unsigned16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

Ce type est un nombre binaire. Le bit de poids le plus fort de l'octet de poids le plus fort est toujours utilisé comme bit de poids le plus fort du nombre binaire; aucun bit de signe n'est inclus. Ce type Unsigned (non signé) a une longueur de deux octets.

5.3.1.4.3.4 UINT

Ce type de l'IEC 61131-3 est le même que le type Unsigned16.

5.3.1.4.3.5 Unsigned32

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	7
2	Data type Name	=	Unsigned32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Ce type est un nombre binaire. Le bit de poids le plus fort de l'octet de poids le plus fort est toujours utilisé comme bit de poids le plus fort du nombre binaire; aucun bit de signe n'est inclus. Ce type Unsigned a une longueur de quatre octets.

5.3.1.4.3.6 UDINT

Ce type de l'IEC 61131-3 est le même que le type Unsigned32.

5.3.1.4.3.7 Unsigned64

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	56
2	Data type Name	=	Unsigned64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

Ce type est un nombre binaire. Le bit de poids le plus fort de l'octet de poids le plus fort est toujours utilisé comme bit de poids le plus fort du nombre binaire; aucun bit de signe n'est inclus. Ce type Unsigned a une longueur de huit octets.

5.3.1.4.3.8 ULINT

Ce type de l'IEC 61131-3 est le même que le type Unsigned64.

5.3.1.5 Types Time (temps)

5.3.1.5.1 TIME

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Nom du type de données	=	TIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Ce type de l'IEC 61131-3 est un nombre binaire en complément à deux d'une longueur de quatre octets. L'unité de temps pour ce type est de 1 ms.

5.3.1.5.2 ITIME

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Nom du type de données	=	ITIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

Cette extension de type de l'IEC 61131-3 est un nombre binaire en complément à deux d'une longueur de deux octets. L'unité de temps pour ce type est de 1 ms.

5.3.1.5.3 FTIME

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Nom du type de données	=	FTIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

Cette extension de type de l'IEC 61131-3 est un nombre binaire en complément à deux d'une longueur de quatre octets. L'unité de temps pour ce type est de 1 µs.

5.3.1.5.4 LTIME

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Nom du type de données	=	LTIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

Cette extension de type de l'IEC 61131-3 est un nombre binaire en complément à deux d'une longueur de huit octets. L'unité de temps pour ce type est de 1 µs.

5.3.2 Types String (chaîne)

5.3.2.1 BitString

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	14
2	Data type Name	=	Bitstring
3	Format	=	STRING
5.1	Octet Length	=	1 à n

Ce type String est défini comme une série d'éléments BitString8.

5.3.2.2 OctetString

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	10
2	Data type Name	=	OctetString
3	Format	=	STRING
4.1	Octet Length	=	1 à n

Un OctetString est une séquence ordonnée d'octets, numérotés de 1 à n. Pour les besoins du débat, l'octet 1 de la séquence est appelé le premier octet. L'IEC 61158-6-2 définit l'ordre d'émission.

5.3.2.3 EPATH

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Nom du type de données	=	EPATH
3	Format	=	STRING
4.1	Octet Length	=	1 à n

Un EPATH est une séquence ordonnée d'octets, numérotés de 1 à n. Son format est spécifié plus précisément dans le paragraphe 4.1.9 de l'IEC 61158-6-2.

5.3.2.4 ETH_MAC_ADDR

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Nom du type de données	=	ETH_MAC_ADDR
3	Format	=	TABLEAU
6.1	Nombre d'éléments de tableau	=	6
6.2	Type de données d'élément de tableau	=	UINT

L'adresse MAC Ethernet est un tableau de 6 octets. Le format d'affichage recommandé est "XX-XX-XX-XX-XX-XX" en commençant par le premier octet. L'adresse MAC Ethernet est attribuée par le fabricant et doit être unique conformément aux exigences de l'ISO/IEC 8802-3. L'exigence générale est que la valeur de ce type doit être des adresses MAC Ethernet utilisées dans les trames Ethernet.

5.3.3 Types Structure

5.3.3.1 DATE_AND_TIME

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	DATE_AND_TIME
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Time_Of_Day_Element
5.2.2	Field Data type	=	TIME_OF_DAY
5.3.1	Field Name	=	Date_Element
5.3.2	Field Data type	=	DATE

Cette extension de type de l'IEC 61131-3 est une structure qui exprime aussi bien la date (nombre de jours du 1972-01-01 jusqu'au 2151-06-06) que l'heure sous la forme d'un nombre de ms à partir de minuit.

5.3.3.2 SHORT_STRING

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	SHORT_STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data type	=	OctetString

Cette extension de type de l'IEC 61131-3 est constituée de deux éléments. Charcount_Element donne le nombre courant de caractères dans Stringcontents_Element (un octet par caractère). Les caractères doivent être tels que spécifiés dans l'ISO/IEC 8859-1.

5.3.3.3 STRING

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Nom du type de données	=	STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	UINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data type	=	OctetString

Ce type de l'IEC 61131-3 est constitué de deux éléments. Charcount_Element donne le nombre courant de caractères dans Stringcontents_Element (un USINT par caractère). Les caractères doivent être tels que spécifiés dans l'ISO/IEC 8859-1.

5.3.3.4 STRING2

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	STRING2
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	UINT
5.3.1	Field Name	=	String2contents_Element
5.3.2	Field Data type	=	OctetString

Cette extension de type de données de l'IEC 61131-3 est constituée de deux éléments. Charcount_Element donne le nombre courant de caractères dans String2contents_Element (un UINT par caractère). Les caractères doivent être tels que spécifiés dans l'ISO/IEC 10646.

5.3.3.5 STRINGN

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	STRINGN
3	Format	=	STRUCTURE
5.1	Number of Fields	=	3
5.2.1	Field Name	=	Charsize_Element
5.2.2	Field Data type	=	UINT
5.3.1	Field Name	=	Charcount_Element
5.3.2	Field Data type	=	UINT
5.4.1	Field Name	=	StringNcontents_Element
5.4.2	Field Data type	=	OctetString

Cette extension de type de l'IEC 61131-3 est constituée de trois éléments. Charsize_Element donne la taille d'un caractère dans StringNcontents_Element (N = nombre de USINT). Charcount_Element donne le nombre courant de caractères dans StringNcontents_Element (N USINT par caractère). Les caractères doivent être tels que spécifiés dans l'ISO/IEC 10646.

5.3.3.6 STRINGI

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	STRINGI
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Stringnum_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	International_String_Array
5.3.2	Field Data type	=	STRINGI_ARRAY

Cette extension de type de l'IEC 61131-3 permet d'utiliser plusieurs représentations de langue pour une même chaîne. Il s'agit d'un type de données structuré qui alloue une variable USINT (Stringnum_Element) contenant le nombre de chaînes de caractères rendus internationaux et une matrice de structures (International_String_Array) contenant les chaînes de caractères rendus internationaux.

La structure des chaînes de caractères internationaux (STRINGI_ELEMENT) est définie comme suit:

- un USINT (Language1_Element) indiquant le premier caractère ASCII de la langue ISO 639-2/T;
- un USINT (Language2_Element) indiquant le deuxième caractère de la langue ISO 639-2/T;
- un USINT (Language3_Element) indiquant le troisième caractère de la langue ISO 639-2/T;
- un USINT (Datatype_Element), limité aux valeurs 0xD0 (STRING), 0xD5 (STRING2), 0xD9 (STRINGN) et 0xDA (SHORT_STRING), indiquant la structure de la chaîne de caractères;
- un UINT (Charset_Element) indiquant le jeu de caractères sur lequel est basée la chaîne de caractères;
- une matrice d'éléments octets qui est la chaîne effective de caractères internationaux (dont le type de données est spécifié dans Datatype_Element).

Les trois caractères pour la langue proviennent de l'ISO 639-2/T (Code terminologique Alpha-3) tandis que les valeurs du jeu de caractères proviennent des codes d'imprimantes IANA MIB (IETF RFC 1759). Les valeurs du jeu de caractères se limitent à celles qui sont données dans le Tableau 1.

Tableau 1 – Codes d'imprimantes IANA MIB valides pour sélection de jeu de caractères

Jeu de caractères	Valeur
ISO 8859-1:1987	4
ISO 8859-2:1987	5
ISO 8859-3:1988	6
ISO 8859-4:1988	7
ISO 8859-5:1988	8
ISO 8859-6:1987	9
ISO 8859-7:1987	10
ISO 8859-8:1989	11
ISO 8859-9:1989	12
-UCS-2 de l'ISO/IEC 10646	1 000
-UCS-4 de l'ISO/IEC 10646	1 001

5.3.3.7 SHORT_STRING

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	SHORT_STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data type	=	OctetString

Cette extension de type de l'IEC 61131-3 est constituée d'un seul élément. Charcount_Element donne le nombre courant de caractères dans Stringcontents_Element (un octet par caractère). Les caractères doivent être tels que spécifiés dans l'ISO/IEC 8859-1.

5.3.3.8 STRINGI_ELEMENT

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	STRINGI_ELEMENT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	6
5.2.1	Field Name	=	Language1_Element
5.2.2	Field Data type	=	USINT
5.3.1	Field Name	=	Language2_Element
5.3.2	Field Data type	=	USINT
5.4.1	Field Name	=	Language3_Element
5.4.2	Field Data type	=	USINT
5.5.1	Field Name	=	Datatype_Element
5.5.2	Field Data type	=	EPATH
5.6.1	Field Name	=	Charset_Element
5.6.2	Field Data type	=	UINT
5.7.1	Field Name	=	Stringcontents _Element
5.7.2	Field Data type	=	SHORT_STRING | STRING | STRING2 | STRINGN

5.4 Spécification de service de l'élément ASE de type de données

Il n'y a pas de services opérationnels définis pour l'objet de type.

6 Spécification de modèle de communication

6.1 Concepts

6.1.1 Généralités

Il y a un système de communication en série pour la communication entre des appareils qui souhaitent échanger de l'information d'application tant du type à temps critique que du type messagerie. Ces appareils comprennent aussi bien des appareils E/S simples (tels que capteurs/actionneurs) que des dispositifs de commande complexes (tels que robots, automates programmables, soudeurs, dispositifs de commande de processus).

Contrairement à certains systèmes de communication d'usage général qui reposent sur le modèle de livraison à destination, ce réseau utilise une variante du modèle push éditeur/abonné appelée modèle producteur/consommateur. Le modèle producteur/consommateur permet l'échange d'informations d'application à temps critique entre un appareil expéditeur (à savoir le producteur) et plusieurs appareils récepteurs (à savoir les consommateurs) sans la nécessité d'envoyer séparément les données vers chaque destination. Cela s'accomplit en joignant un identificateur propre à chaque élément d'informations d'application qui est produit sur le support réseau. Tout appareil qui exige un élément d'informations d'application filtre simplement les données sur le support réseau pour l'identificateur approprié. Plusieurs appareils peuvent recevoir le même élément d'informations d'application issu d'un seul appareil producteur.

La couche application de type 2 peut être associée à plusieurs couches Liaison selon le profil de communication sélectionné.

La couche Liaison de données spécifique de type 2 assure un haut degré d'efficacité de protocoles en utilisant un mécanisme correspondant de transmission de jeton. Ce mécanisme procure à tous les appareils un égal accès au réseau sans le surdébit réseau associé à la transmission d'un jeton à chaque appareil lui accordant la permission d'envoyer des données. Le protocole utilise un mécanisme de programmation basé sur le temps qui donne aux appareils de réseau l'accès déterministe et prévisible au support tout en prévenant les collisions dans le réseau. Ce mécanisme de programmation permet que des données à temps critiques, qui sont requises sur une base périodique, répétable et prévisible, soient produites selon un échéancier prédéfini sans la perte d'efficacité associée à la demande ou au "sondage" en continu des données requises. Le protocole prend en charge un mécanisme complémentaire qui permet à des données qui ne sont pas à temps critique par nature ou qui sont seulement requises sur une base occasionnelle d'utiliser tout temps-réseau disponible. Ces données non programmées sont émises après l'achèvement de la production des données à temps critique et avant le début de la prochaine production programmée de données à temps critique.

6.1.2 Concepts généraux

La plupart des concepts généraux décrits à l'Article 4 s'appliquent, avec un certain nombre de restrictions ou d'additions telles que notées:

- la couche application inclut les fonctionnalités issues des couches intermédiaires qui sont nécessaires au mapping correct avec la couche Liaison de données;
- les relations Client/Serveur peuvent être un à un, mais peuvent aussi être un à plusieurs (voir le modèle d'AR en 6.3.1);
- une variante du module Push Editeur/Abonné, à savoir le modèle Producteur/Consommateur, est utilisée comme la base de toutes les AR (voir le modèle d'AR en 6.3.1);
- une vue d'ensemble des types ASE/APO et de leurs relations est donnée en 6.1.3;
- les AR suivent un modèle qui est détaillé en 6.3.1;
- une désignation et un adressage spécifiques sont spécifiés en 6.1.4;
- les attributs et paramètres FAL communs énumérés dans le paragraphe 9.7 de l'IEC 61158-1 ne sont pas utilisés; au contraire, ils sont spécifiés dans les paragraphes spécifiques au Type correspondant;
- il n'y a pas de services Abort ou Reject, il n'y a que des confirmations de résultats négatifs et, donc, la procédure d'erreur décrite dans l'Annexe A ne s'applique pas.

6.1.3 Relations entre les ASE

Chaque nœud doit contenir au minimum l'instance 1 des classes d'objets ASE suivantes dans sa couche application.

ASE Object Management (gestion d'objets):

- Identity ("Identité", informations d'identification et générales relatives à l'appareil)
- Message Router (Routeur de message, point de connexion de messagerie pour les communications au sein de l'appareil)

ASE Connection Manager (gestionnaire de connexion):

- Connection Manager (établissement et maintenance de connexions)

ou ASE Connection (Connexion):

- Connection (point de connexion de messagerie pour les communications au sein de l'appareil)

ASE AR:

- Unconnected Message Manager ("Gestionnaire de messages non connectés", services de messages en file d'attente sur une seule liaison, facultatif pour CP 2/3)

D'autres objets de couche application peuvent être mis en œuvre dans un certain nombre de nœuds seulement:

ASE Object Management (gestion d'objets):

- Objet Assembly (lie des données issues de plusieurs objets devant être envoyées par une connexion)
- Objet Acknowledge Handler (gère les messages d'acquittement issus des objets d'application)
- Objet Time Sync (interface au protocole de synchronisation d'horloge de précision selon l'IEC 61588:2009)
- Objet Parameter (interface publique aux données de configuration des appareils)
- d'autres objets spécifiques à une application construits selon le modèle formel général de type 2

ASE AR:

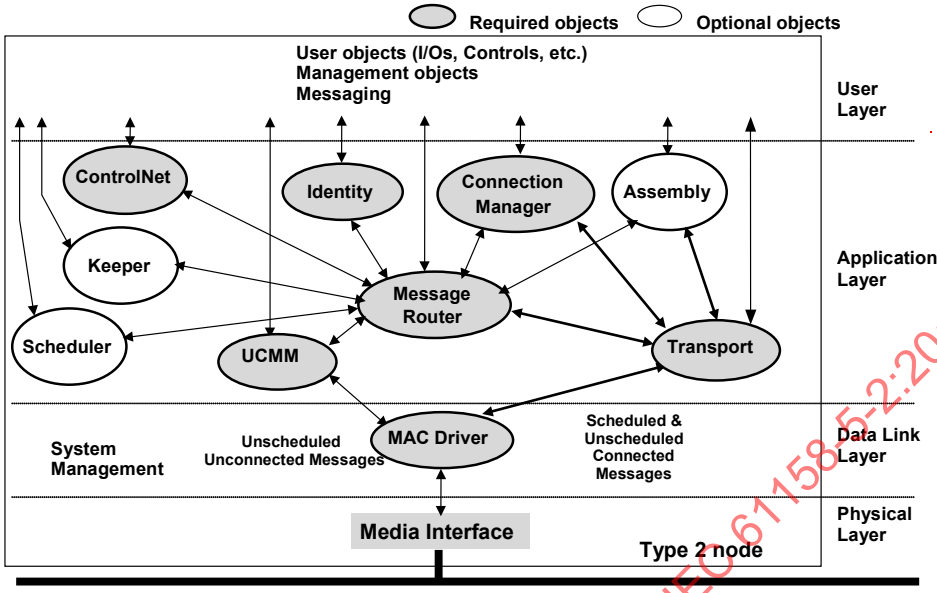
- Transports (services connectés de messagerie et de données).

Bien d'autres objets appartiennent à la gestion de système, principalement concernés par la couche Liaison de données:

- Objet ControlNet (interfaces de gestion de poste aux couches inférieures, requises dans tous les appareils utilisant la couche Liaison de données de type 2),
- Objet Keeper (détient et distribue des attributs de tous les appareils sur la liaison, un étant requis par liaison, avec une ou plusieurs sauvegardes facultatives – utilisé conjointement avec la couche Liaison de données de type 2),
- Objet Scheduling (détient des informations sur les programmes de liaison, un étant requis dans chaque émetteur de connexion – utilisé conjointement avec la couche Liaison de données de type 2),
- Objet TCP/IP Interface (fournit le mécanisme pour configurer l'interface TCP/IP d'un appareil, requis dans tous les appareils avec une interface TCP/IP),
- Objet Ethernet Link (maintient des compteurs et des informations de statut spécifiques à une liaison pour un port Ethernet, requis dans tous les appareils avec un port Ethernet),
- Objet DeviceNet (interfaces de gestion de poste aux couches inférieures, requises dans tous les appareils utilisant la couche Liaison de données selon l'ISO 11898:1993),
- Objet Connection Configuration (interface pour créer, configurer et commander des connexions dans un appareil),
- Objet DLR (interface de configuration et d'informations de statut pour le protocole DLR),
- Objet QoS (interface de configuration pour les comportements relatifs à la qualité de service dans les appareils),
- Objet Port (décrit les ports de type 2 présents sur l'appareil).
- Objet de protocole PRP/HSR (interface de configuration et de diagnostic pour les protocoles PRP et HSR),
- Objet de table de nœuds PRP/HSR (enregistrement de tous les nœuds compatibles PRP ou HSR détectés sur le réseau).

NOTE Ces objets de gestion de couches inférieures sont décrits dans l'IEC 61158-4-2.

Les interactions des ASE et d'objets pour un appareil utilisant les protocoles de type 2 de couches tant Application que Liaison de données sont représentées à la Figure 1:

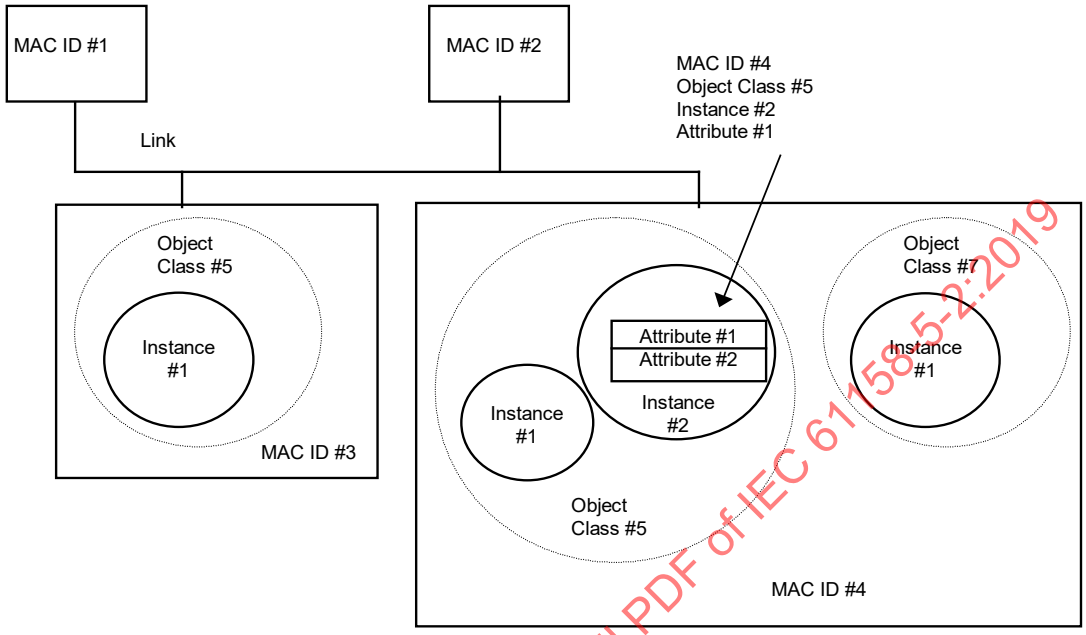


Anglais	Français
Required Objects	Objets requis
Optional Objects	Objets facultatifs
User objects (I/Os, Controls, etc.)	Objets utilisateur (E/S, Commandes, etc.)
Management objects	Objets de gestion
Messaging	Messagerie
User Layer	Couche Utilisateur
ControlNet	ControlNet
Identity	Identity
Connection Manager	Connection Manager
Assembly	Assembly
Keeper	Keeper
Message Router	Message Router
Scheduler	Scheduler
UCMM	UCMM
Transport	Transport
MAC Driver	Pilote MAC
Application Layer	Couche application
Data Link Layer	Couche liaison de données
System Management	Gestion de système
Unscheduled Unconnected Messages	Messages non connecté et non programmé
Scheduled & Unscheduled Connected Messages	Messages connectés programmés et non programmés
Physical Layer	Couche physique
Media Interface	Interface aux supports
Type 2 Node	Nœud de type 2

Figure 1 – Vue d'ensemble des ASE et des classes d'objets

6.1.4 Dénomination et adressage

Les informations présentées ici donnent une base commune pour adresser logiquement des composants physiques distincts à travers le réseau. Ces termes d'adressage sont utilisés dans les spécifications. Il convient de faire référence à la Figure 2 dans le débat qui suit.



Anglais	Français
MAC ID	Identificateur MAC
Object Class	Classe d'objets
Instance	Instance
Attribute	Attribut
Link	Liaison

Figure 2 – Format d'adressage utilisant les ID de MAC, de classe, d'instance et d'attribut

Le terme "nœud" est utilisé pour se référer à la partie d'un appareil qui contient une interface de liaison et répond à un seul MAC ID (identificateur MAC). Le terme "appareil" se réfère à un appareil physique complet se raccordant au réseau. Un appareil est capable de contenir plusieurs nœuds.

NOTE Si la couche application de type 2 est utilisée conjointement avec Ethernet, le MAC ID référencé correspond à l'adresse IP, et non au MAC ID Ethernet.

Le Class ID (identificateur de classe) est une unique valeur d'identification entière attribuée à chaque classe d'objets accessible à partir du réseau. La classe d'objets peut être référencée par son Class ID (identificateur de classe). Dans toutes les spécifications IEC 61158 pour le Type 2, le code de classe est synonyme de Class ID.

L'Instance ID (identificateur d'instance) est une valeur d'identification entière attribuée à une instance d'objet lorsqu'elle est créée, qui l'identifie parmi toutes les Instances de la même Classe. Cet entier est unique au sein du <Node:Class> dans lequel il réside.

L'Attribute ID (identificateur d'attribut) est une valeur d'identification entière qui est unique parmi tous les autres attributs de l'objet en question.

L'adresse de l'attribut numéro 1 de l'instance 2 dans la classe 5 dans le nœud avec un MAC ID de 4 est MAC ID#4: Object Class #5: Instance #2: Attribute #1 comme indiqué à la Figure 2. Cette nomenclature est appelée adressage Classe/Instance/Attribut.

Des informations relatives à la façon d'adresser un objet à travers plusieurs liaisons ou en utilisant un nom sont données dans l'IEC 61158-6-2 (Définition de chemin).

6.1.5 Types de données

6.1.5.1 Spécification générale des types de données

La spécification d'un type de données est constituée de la plage de valeurs que les variables du type peuvent prendre et les opérations exécutées sur ces variables.

Les données sont constituées de types de données (primitives) élémentaires. Ces types de données élémentaires sont utilisés pour construire les types de données (construits) dérivés.

NOTE 1 Les spécifications des types de données élémentaires et dérivés correspondent à la notation de l'IEC 61131-3. En outre, sachant que les schémas fonctionnels tels que définis dans l'IEC 61131-3 ont une structure de données associée et aussi un ensemble d'opérations définies, ces éléments sont également spécifiés ci-dessous comme des types de données supplémentaires.

Les types de données dérivés sont les suivants:

dérivé directement,

énuméré,

sous-plage,

structuré,

matrice.

NOTE 2 Les types de données dérivés sont définis dans l'IEC 61131-3:2003, en 1.3 et 2.3.3. Le moyen de spécifier ces types de données et leurs valeurs initiales par défaut est défini dans l'IEC 61131-3:2003, paragraphes 2.3.3.1 et 2.3.3.2. L'usage des variables de ces types de données est défini dans l'IEC 61131-3:2003, en 2.3.3.3.

6.1.5.2 Types de données élémentaires pris en charge

Les types de données de base suivants définis à l'Article 5 sont pris en charge:

BOOL
SINT
INT
DINT
LINT
USINT
UINT
UDINT
ULINT
REAL
LREAL
ITIME
TIME
FTIME
LTIME

DATE
TIME_OF_DAY ou TOD
DATE_AND_TIME ou DT
STRING
STRING2
STRINGN
STRINGI
SHORT_STRING
SWORD
WORD
DWORD
LWORD
EPATH

6.1.5.3 Conformité à l'IEC 61131-3

6.1.5.3.1 Enoncé de conformité

NOTE 1 Le paragraphe 6.1.5.3 donne des informations en rapport avec les seuls types de données et opérations associées qui sont définis dans la présente partie de l'IEC 61158.

NOTE 2 l'IEC 61131-3:2003, paragraphe 1.5.1, définit les exigences qui sont satisfaites par les systèmes d'automates programmables revendiquant la conformité à l'IEC 61131-3. Cela fournit les informations relatives au type de données à inclure dans la documentation d'unités fonctionnelles qui prennent en charge les types de données définis dans la présente partie de l'IEC 61158. Des sous-ensembles ou extensions de cette documentation sont donnés selon le cas approprié à l'unité fonctionnelle conforme spécifique.

Une mise en œuvre doit se conformer aux exigences de l'IEC 61131-3 pour les caractéristiques de langue telles que spécifiées du Tableau 2 au Tableau 4.

Tableau 2 – Eléments communs

IEC 61131-3:2003 Tableau/ caractéristique	Description de la caractéristique
10/1	Type de données BOOL
10/2	Type de données SINT
10/3	Type de données INT
10/4	Type de données DINT
10/5	Type de données LINT
10/6	Type de données USINT
10/7	Type de données UINT
10/8	Type de données UDINT
10/9	Type de données ULINT
10/10	Type de données REAL
10/11	Type de données LREAL
10/12	Type de données TIME
10/13	Type de données DATE
10/14	Type de données TIME_OF_DAY ou TOD
10/15	Type de données DATE_AND_TIME ou DT
10/16	Type de données STRING
10/17	Type de données SWORD

IEC 61131-3:2003 Tableau/ caractéristique	Description de la caractéristique
10/18	Type de données WORD
10/19	Type de données DWORD
10/20	Type de données LWORD
12/1	Types de données dérivés directement
12/2	Types de données énumérés (Enumerated)
12/3	Types de données sous-plage (Subrange)
12/4	Types de données matrice (Array)
12/5	Types de données structurés
13	Valeurs initiales par défaut normalisées
paragraphe 2.5.1.3	Fonctions déclarées par l'utilisateur (aucune entrée de tableau)
22/1	Conversions de type (voir Tableau 4)
22/2	Fonction TRUNC
22/3	Fonctions BCD_TO_**
22/4	Fonctions *_TO_BCD
23/1-11	Fonctions normalisées d'une variable numérique: ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN
24/12n-18n	Fonctions arithmétiques nommées normalisées: ADD, MUL, SUB, DIV, MOD, EXPT, MOVE
24/12s-15s, 17s,18s	Fonctions arithmétiques symboliques normalisées: +, *, –, /, **,:=
25/1-4	Fonctions normalisées de chaîne de bits: SHL, SHR, ROR, ROL
26/5s-8s	Fonctions booléennes nommées normalisées au niveau du bit: AND, OR, XOR, NOT
26/5s-7s	Fonctions booléennes symboliques normalisées au niveau du bit: &, >=1, =2k+1
27/1-4	Fonctions normalisées de sélection: SEL, MAX, MIN, LIMIT, MUX
28/5n-10n	Fonctions normalisées de comparaison nommées: GT, GE, EQ, LE, LT, NE
28/5s-10s	Fonctions normalisées de comparaison symboliques: >, >=, =, <=, <, <>
29/1-9	Fonctions normalisées de chaîne de caractères: LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND
30/1-14	Fonctions normalisées de types de données de temps: (voir NOTE) ADD, SUB, MUL, DIV, CONCAT, DATE_AND_TIME_TO_TIME_OF_DAY, TIME_OF_DAY_TO_DATE_AND_TIME
31/1-4	Fonctions normalisées de types de données énumérés: SEL, MUX, EQ, NE
32	Mécanismes d'accès normalisés à des paramètres E/S de schéma fonctionnel
33/8a,8b,9a,9b	Schémas fonctionnels définis par l'utilisateur selon l'IEC 61131-3:2003, paragraphe 2.5.2.2, avec options d'entrées graphiques ou textuelles de front montant ou descendant
34/1-3	Schémas fonctionnels bistables normalisés: SR, RS, SEMA
35/1,2	Schémas fonctionnels normalisés de détection de bord: R_EDGE, F_EDGE
36/1-3	Schémas fonctionnels normalisés de compteurs: CTU, CTD, CTUD
37/1,2a,3a, 4	Schémas fonctionnels normalisés de temporisateurs: TP, TON, TOF, RTC
55/1-17	Opérateurs normalisés: (), évaluation de fonction, **,–, NOT, *, /, MOD, +,–, <, >, <=, >=, =, <>, &, AND, XOR, OR

NOTE l'IEC 61131-3:2003, Tableau 30, limite les types de données auxquels ces opérations s'appliquent.

Tableau 3 – Eléments de langue ST

IEC 61131-3:2003 Tableau/ caractéristique	Description de la caractéristique
55/1-17	Opérateurs normalisés: (), évaluation de fonction, **,–, NOT, *, /, MOD, +,–, <, >, <=, >=, =, <>, &, AND, XOR, OR
56/2	Invocation de schéma fonctionnel et usage de la sortie

Tableau 4 – Opérations de conversion de type

Opération	Result	Conditions d'erreur
ANY_BIT_TO_ANY_BIT	Voir NOTE 4	Aucune
ANY_BIT_TO_ANY_INT	OUTmin + Sbk2k (NOTE 5)	Résultat > OUTmax
ANY_BIT_TO_BOOL	FALSE si IN = 0 TRUE sinon	Aucune
ANY_BIT_TO_STRING	Voir NOTE 6	Aucune
ANY_DATE_TO_STRING	Voir NOTE 7	Aucune
ANY_INT_TO_BOOL	FALSE si IN = 0 TRUE sinon	Aucune
ANY_NUM_TO_ANY_INT	IN (NOTE 8)	(IN > OUTmax) ou (IN < OUTmin)
ANY_NUM_TO_ANY_REAL	IN	Voir NOTE 9
ANY_NUM_TO_DATE	Voir NOTE 10	
ANY_NUM_TO_STRING	Voir NOTE 11	
ANY_NUM_TO_TIME	Voir NOTE 12	
ANY_NUM_TO_TOD	Voir NOTE 13	
ANY_REAL_TO_BOOL	FALSE si IN = 0,0 TRUE sinon	Aucune
BOOL_TO_ANY_BIT BOOL_TO_ANY_INT	0 si IN = FALSE 1 si IN = TRUE	Aucune
BOOL_TO_ANY_REAL	0,0 si IN = FALSE 1,0 si IN = TRUE	Aucune
BOOL_TO_STRING	'FALSE' si IN = FALSE 'TRUE' si IN = TRUE	Aucune
DATE_TO_ANY_NUM	Voir NOTE 14	Résultat > OUTmax
STRING_TO_ANY	Données converties	Voir NOTE 15
TIME_TO_ANY_NUM	Voir NOTE 16	Résultat > OUTmax
TOD_TO_ANY_NUM	Voir NOTE 17	Résultat > OUTmax

NOTE 1 L'utilisation des types de données génériques ANY_NUM, ANY_REAL, ANY_INT, et ANY_BIT définis dans le paragraphe 2.3.2 de l'IEC 61131-3:2003, permet d'impliquer une famille de conversions. Par exemple, la conversion BOOL_TO_ANY_REAL vise à sous-entendre BOOL_TO_REAL et BOOL_TO_LREAL.

NOTE 2 IN renvoie à la valeur de la variable d'entrée de la fonction de conversion de type.

NOTE 3 OUT_{min} et OUT_{max} renvoient aux valeurs minimale et maximale du type de données de sortie de la fonction de conversion, telle que définie à l'Article 5.

NOTE 4 Dans les conversions des types bitstring (chaîne de bits), si le nombre de bits dans la variable d'entrée IN est inférieur au nombre de bits dans la variable de sortie OUT, les bits de l'entrée sont copiés dans les bits de poids faible correspondants du résultat, le reste du résultat étant rempli de zéros. Si le nombre de bits de IN est supérieur au nombre de bits de OUT, les bits de poids faible de IN sont copiés dans les bits correspondants du résultat. Par exemple:

SWORD_TO_WORD(16#FF) = 16#00FF
et
WORD_TO_SWORD (16#0FF0) = 16#F0

NOTE 5 La numérotation des bits dans cette expression est telle que spécifiée dans l'IEC 61158-6-2.

NOTE 6 Le résultat de conversion d'une variable du type bitstring en type STRING consiste en une chaîne contenant la représentation littérale de base 16 de la valeur de la variable, telle que définie dans l'IEC 61131-3:2003, paragraphe 2.2.1, en des caractères issus du jeu de caractères de l'ISO/IEC 646.

NOTE 7 Le résultat de conversion d'une variable de type date et/ou time of day en type STRING consiste en une chaîne contenant la représentation littérale de la valeur de la variable, telle que définie dans l'IEC 61131-3:2003, paragraphe 2.2.1, en des caractères issus du jeu de caractères de l'ISO/IEC 646.

NOTE 8 La conversion de REAL et LREAL en des types entiers est accomplie par arrondi tel que spécifié dans l'ISO/IEC/IEEE 60559.

NOTE 9 Des erreurs d'arrondi peuvent se produire si le nombre de bits significatifs dans la variable d'entrée est supérieur au nombre d bis significatifs dans la représentation en virgule flottante de la sortie. De même, des erreurs d'étendue du type noté pour ANY_NUM_TO_ANY_INT peuvent apparaître dans LREAL_TO_REAL.

NOTE 10 La conversion d'une variable d'un type numérique en type DATE a le même résultat que la conversion de la variable en UINT, le résultat étant interprété comme le nombre de jours à partir de 1972-01-01.

NOTE 11 La conversion d'une variable d'un type numérique en type STRING consiste en une chaîne contenant la représentation littérale de la valeur de la variable, telle que définie dans l'IEC 61131-3:2003, paragraphe 2.2.1, en des caractères issus du jeu de caractères de l'ISO/IEC 646.

NOTE 12 La conversion d'une variable d'un type numérique en type TIME a le même résultat que la conversion de la variable en DINT, le résultat étant interprété comme une durée en millisecondes.

NOTE 13 La conversion d'une variable d'un type numérique en type TOD a le même résultat que la conversion de la variable en UDINT, le résultat étant interprété comme un temps en millisecondes à partir de minuit.

NOTE 14 La conversion d'une variable du type DATE en un type numérique est la même que la conversion d'une variable du type UINT en le type numérique correspondant, le résultat étant l'équivalent numérique des jours à partir de 1972-01-01.

NOTE 15 Il y a erreur si les données STRING à convertir ne sont pas dans le format pour la représentation externe du type de données de la sortie tel que spécifié dans l'IEC 61131-3:2003, paragraphe 2.2 ou si le résultat de la conversion se situe hors de l'étendue $\{OUT_{min}.OUT_{max}\}$.

NOTE 16 La conversion d'une variable du type TIME en un type numérique est la même que la conversion d'une variable du type DINT en le type numérique correspondant, le résultat étant l'équivalent numérique de l'intervalle de temps correspondant exprimé en millisecondes.

NOTE 17 La conversion d'une variable du type TOD (TIME_OF_DAY) en un type numérique est la même que la conversion d'une variable du type UDINT en le type numérique correspondant, le résultat étant l'équivalent numérique du temps à partir de minuit exprimé en millisecondes.

6.1.5.3.2 Paramètres dépendant de la mise en œuvre

Les valeurs de paramètres dépendant de la mise en œuvre issues de l'IEC 61131-3:2003, Tableau D.1, sont telles que montrées dans le Tableau 5.

NOTE Les valeurs d'autres paramètres dépendant de la mise en œuvre sont définies dans d'autres normes ou dans les spécifications des unités fonctionnelles individuelles selon le cas.

Tableau 5 – Valeurs des paramètres dépendant de la mise en œuvre

Paragraphe de l'IEC 61131-3:2003	Paramètre	Valeur
2.2.3.1	Etendue des valeurs de durée	La même que LINT en microsecondes
2.3.1	Etendue des valeurs de variables du type TIME	La même que DINT en millisecondes
	Précision de représentation de secondes dans les types TIME_OF_DAY et DATE_AND_TIME	1 ms
2.3.3.1	Nombre maximal de valeurs énumérées	256
2.3.3.2	Longueur maximale par défaut des variables STRING	256
	Longueur admissible par défaut des variables STRING	65 536
2.4.1.2	Nombre maximal d'indices	8
	Etendue maximale de valeurs d'indice	0 – 255
	Nombre maximal de niveaux des structures	8
2.5.1.5	Entrées maximales des fonctions extensibles	8
2.5.1.5.1	Effets des conversions de type sur l'exactitude	Comme défini dans le Tableau 4
2.5.1.5.2	Exactitude des fonctions à une variable	Comme défini dans l'ISO/IEC/IEEE 60559
2.5.2.3.3	PVmin, PVmax de compteurs	0, 65 535

6.1.5.3.3 Extensions de langage

Les extensions à l'IEC 61131-3 définies dans la présente partie sont énumérées dans le Tableau 6. Lorsque ces extensions sont utilisées dans un appareil particulier, les références de paragraphes dans ce tableau doivent être remplacées par les références aux descriptions des extensions correspondantes dans la documentation de l'unité fonctionnelle.

Tableau 6 – Extensions à l'IEC 61131-3:2003

Paragraphe	Description
2.1.1	Type de données ITIME Type de données FTIME Type de données LTIME Type de données STRING2 Type de données STRINGN Type de données SHORT_STRING Type de données STRINGI Type de données EPATH
2.1.4	Types chaîne de bits structurés
2.2	Opérations sur des variables STRING2
2.2.4.1	Accès chaîne de bit numéroté
2.2.4.2	Accès chaîne de bits structuré

6.2 Eléments ASE

6.2.1 ASE Object Management (gestion d'objets)

6.2.1.1 Vue d'ensemble

L'ASE Object Management de la FAL sert à accéder à tous les éléments d'application ou éléments de gestion de système accessibles par le réseau.

Une règle d'accès est associée à chaque attribut d'une classe d'objets; elle spécifie comment un demandeur peut accéder à l'attribut. Les définitions des règles d'accès se présentent comme suit:

- Définissable (Set) – Il doit être accédé à l'attribut par au moins un des services Set (pour les attributs énumérés, si l'appareil ne prend en charge qu'une seule des valeurs et que la définition d'objet n'exige pas plus d'une valeur à prendre en charge, alors l'attribut peut être mis en œuvre comme Get seulement);

  sauf spécification contraire dans la définition d'objet, les attributs Settable doivent également être Gettable et peuvent être accessibles par les services Get;

- Gettable (Get) <Récupérable (Récupérer)>: on doit accéder à l'attribut par au moins un des services GET.

Certains attributs peuvent nécessiter l'utilisation de la mémoire non volatile afin de maintenir leurs valeurs d'un cycle de puissance à l'autre. Le cas échéant, ceci est spécifié comme suit dans les définitions d'objet:

- non volatile (NV) indique que la valeur doit être sauvegardée;
- volatile (V) indique que la valeur ne doit pas être sauvegardée.

6.2.1.2 Spécification de classe du modèle de gestion de FAL

6.2.1.2.1 Modèle formel général

6.2.1.2.1.1 Définition de classe

La classe AP de base ci-dessous spécifie les attributs et services communs définis pour toutes les autres classes AP, y compris ceux définis par l'utilisateur (spécifiques à une application).

Placé en face d'un attribut ou d'un service, le symbole (*) signifie que cet attribut ou ce service est soit obligatoire, soit facultatif, sur la base de certaines contraintes définies dans la description de l'attribut/du service.

FAL ASE:			**FAL Management ASE**
CLASS:			**Base_Object**
CLASS ID:			**NULL**
PARENT CLASS:			**TOP**
ATTRIBUTS D'ACCES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
ATTRIBUTS DE GESTION DE SYSTEME (ATTRIBUTS DE CLASSE):			
1	(*)	Attribute:	Revision
2	(o)	Attribute:	Max Instance
3	(o)	Attribute:	Number of Instances
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list

5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes

ATTRIBUTS DE GESTION D'OBJETS (ATTRIBUTS D'INSTANCE):

1	(o)	Attribute:	Attr1

SERVICES DE GESTION DE SYSTEME:

1	(o)	Mgt Service:	Get_Attributes_All
2	(o)	Mgt Service:	Set_Attributes_All
3	(o)	Mgt Service:	Get_Attribute_List
4	(o)	Mgt Service:	Set_Attribute_List
5	(o)	Mgt Service:	Reset
6	(o)	Mgt Service:	Start
7	(o)	Mgt Service:	Stop
8	(o)	Mgt Service:	Create
9	(o)	Mgt Service:	Delete
13	(o)	Mgt Service:	Apply_Attributes
14	(o)	Mgt Service:	Get_Attribute_Single
16	(o)	Mgt Service:	Set_Attribute_Single
17	(o)	Mgt Service:	Find_Next_Object_Instance
21	(o)	Mgt Service:	Restore
22	(o)	Mgt Service:	Save
23	(o)	Mgt Service:	NOP
24	(o)	Mgt Service:	Get_Member
25	(o)	Mgt Service:	Set_Member
26	(o)	Mgt Service:	Insert_Member
27	(o)	Mgt Service:	Remove_Member
28	(o)	Mgt Service:	Group_Sync

SERVICES DE GESTION D'OBJETS:

1	(o)	Ops Service:	Get_Attributes_All
2	(o)	Ops Service:	Set_Attributes_All
3	(o)	Ops Service:	Get_Attribute_List
4	(o)	Ops Service:	Set_Attribute_List
5	(o)	Ops Service:	Reset
6	(o)	Ops Service:	Start
7	(o)	Ops Service:	Stop
8	(o)	Ops Service:	Create
9	(o)	Ops Service:	Delete
10	(o)	Ops Service:	Multiple_Service_Packet
13	(o)	Ops Service:	Apply_Attributes
14	(o)	Ops Service:	Get_Attribute_Single
16	(o)	Ops Service:	Set_Attribute_Single
21	(o)	Ops Service:	Restore
22	(o)	Ops Service:	Save
23	(o)	Ops Service:	NOP
24	(o)	Ops Service:	Get_Member
25	(o)	Ops Service:	Set_Member
26	(o)	Ops Service:	Insert_Member
27	(o)	Ops Service:	Remove_Member
28	(o)	Ops Service:	Group_Sync

SERVICES SPECIFIQUES A L'OBJET:

1	(o)	Mgt/Ops Service:	Service1

6.2.1.2.1.2 Attributs d'accès

Ces attributs internes identifient de manière univoque une instance d'objet au sein d'un appareil. Les valeurs correspondantes sont utilisées comme partie du chemin dans des demandes de services pour spécifier l'instance d'objet cible.

Object Instance number

Numéro unique associé à une instance d'objet donnée. Instance number 0 (zéro) signifie que l'accès à la classe d'objets elle-même est demandé.

Symbolic name

Nom facultatif qui peut être associé à une instance d'objet donnée.

6.2.1.2.1.3 Attributs de gestion de système (attributs de classe)

Attributs au niveau classe. Un attribut dont la valeur est partagée par tous les objets au sein d'une même classe est appelé Class Attribute (attribut de classe).

Sept Class Attribute ID (identificateurs d'attribut de classe) prédéfinis sont réservés pour la définition d'objet de classe. Les sept attributs de classe prédéfinis/réservés ont les définitions énumérées ci-dessous. Comme ces attributs sont réservés, les numéros d'ID d'attribut de classe 1 à 7 sont toujours réservés. Par conséquent, si un attribut de classe est ajouté à une spécification d'objets, il doit commencer par AttributeID#8 (c'est-à-dire: ID d'attribut n°8).

Si un Attribut de classe est facultatif, une valeur par défaut ou une méthode de traitement de cas spéciaux doit être définie afin que le Client (Demandeur) puisse traiter le message d'erreur qui se produit lors de l'accès aux objets qui choisissent de ne pas mettre en œuvre l'attribut de classe.

Tous les attributs de classe communs ont une règle d'accès Get seulement.

Revision

Révision de cet objet. La valeur de départ attribuée à cet attribut est un (1). Si des mises à jour nécessitant une augmentation de cette valeur sont effectuées, la valeur de cet attribut augmente de 1.

Si la valeur est 1, cet attribut est facultatif dans des mises en œuvre. Si la valeur est supérieure à 1, cet attribut est obligatoire. L'attribut Revision d'un objet spécifie l'interface à cet objet, qui doit englober la totalité des éléments dans la spécification d'objet, y compris les services, les attributs, les connexions et le comportement.

Max Instance

Numéro d'instance maximal d'un objet actuellement créé à ce niveau de classe dans l'appareil. Le numéro d'instance le plus élevé d'un objet à ce niveau de hiérarchie de classe créé dans l'appareil.

Number of Instances

Nombre d'instances d'objets actuellement créées à ce niveau de classe dans l'appareil. Le nombre d'instances d'objet à ce niveau de hiérarchie de classe.

Optional attribute list

Liste d'attributs d'instance facultatifs utilisés dans une mise en œuvre de classe d'objets. Liste de numéros d'attributs spécifiant les attributs facultatifs mis en œuvre dans l'appareil pour cette classe.

Number of attributes

Nombre d'attributs dans la liste d'attributs facultatifs.

Optional attributes

Liste de numéros d'attributs facultatifs.

Optional service list

Liste de services facultatifs utilisés dans une mise en œuvre de classe d'objets. Liste de codes de service spécifiant les services facultatifs mis en œuvre dans l'appareil pour cette classe.

Si un service facultatif est mis en œuvre dans une classe et l'attribut de classe Optional Service List est également mis en œuvre dans la classe, le service doit être inclus dans l'Optional Service List.

Number services

Nombre de services dans la liste de services facultatifs.

Optional services

Liste de codes de service facultatifs.

Maximum ID Number Class Attributes

Le numéro d'Attribute ID du dernier attribut de classe de la définition de classe mise en œuvre dans l'appareil.

NOTE Il permet de simplifier l'autodétermination de la mise en œuvre de classe par un terminal distant.

Maximum ID Number Instance Attributes

Le numéro d'Attribute ID du dernier attribut d'instance de la définition de classe mise en œuvre dans l'appareil.

NOTE Il permet de simplifier l'autodétermination de la mise en œuvre de classe par un terminal distant.

6.2.1.2.1.4 Attributs de gestion d'objets

Attributs au niveau instance.

Un Attribut d'instance est un attribut dont la valeur est propre à une instance d'objet et dont la définition est partagée par toutes les instances d'un objet. Chaque instance a seulement besoin de prendre en charge les attributs facultatifs qui lui sont applicables. Si une instance ne prend pas en charge un attribut facultatif, l'erreur Attribute Not Supported (code General Status 0x14) doit être retournée pour que les services prennent en charge cet attribut.

Les attributs d'instance sont définis dans les mêmes termes que les attributs de classe. Il n'y a pas d'Instance Attributes réservés.

Instance ID = 0 est un cas spécial. Un service dirigé vers Instance ID = 0 doit être appliqué à la classe, pas à une instance particulière.

Chaque instance a un jeu unique d'attributs. Les Instance attribute ID peuvent être numérotées à partir de 1 sans la moindre corrélation avec les Attribute ID de la classe dont l'objet est une instance. Il n'y a pas d'Instance Attributes réservés.

6.2.1.2.1.5 Services de gestion de système (niveau classe) et de gestion d'objets (niveau instance)

Tous les services communs se comportent de la même façon, qu'ils soient utilisés comme services de gestion de système ou comme services de gestion d'objets.

NOTE 1 Pour invoquer le niveau de classe, spécifiez un Instance ID d'objet de zéro (voir IEC 61158-6-2, 4.1.9.4.1).

Get_Attributes_All

Le service Get_Attribute_All retourne le contenu de tous les attributs de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

Set_Attributes_All

Le service Set_Attribute_All modifie le contenu de tous les attributs de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

Get_Attribute_List

Le service Get_Attribute_List retourne le contenu des attributs récupérables sélectionnés de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

Set_Attribute_List

Le service Set_Attribute_List met à jour le contenu des attributs sélectionnés de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

Reset

Le service Reset invoque le service Reset de la classe ou instance d'objet APO spécifiée.

NOTE 2 Typiquement, cela entraînerait une transition vers un état ou mode par défaut.

Start

Le service Start invoque le service Start de la classe ou instance d'objet APO spécifiée.

NOTE 3 Typiquement, cela placerait une instance d'objet dans un état ou mode "running" (en marche).

Stop

Le service Stop invoque le service Stop de la classe ou instance d'objet APO spécifiée.

NOTE 4 Typiquement, cela placerait une instance d'objet dans un état ou mode "stopped" (arrêté) ou "idle" (repos).

Create

Le service Create a pour résultat l'instanciation d'une nouvelle d'instance d'objet au sein de la classe d'objets spécifiée.

Delete

Le service Delete supprime une instance d'objet de la classe d'objets spécifiée.

Multiple_Service_Packet

Le service Multiple_Service_Packet accomplit un ensemble de services sous la forme d'une séquence autonome.

Apply_Attributes

Le service Apply_Attributes entraîne que les valeurs d'attribut dont l'utilisation est en cours s'utilisent activement dans la classe ou instance d'objets APO spécifiée.

Get_Attribute_Single

Le service Get_Attribute_Single retourne le contenu de l'attribut spécifié de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

Set_Attribute_Single

Le service Set_Attribute_Single modifie le contenu de l'attribut spécifié de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

Find_Next_Object_Instance

Le service Find_Next_Object_Instance doit être pris en charge au niveau de la classe d'objets APO uniquement. Il amène la classe d'objets APO spécifiée à rechercher et retourner une liste d'identificateurs d'instance (Instance ID) associés à des instances d'objets existants. Les objets existants sont ceux qui sont actuellement accessibles à partir de la liaison.

Restore

Le service Restore restaure le contenu des attributs d'une classe ou instance d'objets APO à partir d'un emplacement de stockage accessible par le service Save. Les données d'attribut sont copiées d'une zone de stockage vers une zone mémoire actuellement active utilisée par la classe ou l'instance d'objet.

Save

Le service Save copie le contenu des attributs d'une classe ou instance d'objets APO en un emplacement accessible par le service Restore.

NOP

Le service NOP amène l'instance d'objet APO de réception à retourner une réponse *No Operation* sans accomplir aucune autre action interne.

NOTE 5 Ce service peut être utilisé pour tester si, oui ou non, une classe/instance d'objet particulière est encore présente et répond sans entraîner de changement d'état.

Get_Member

Le service Get_Member renvoie les informations sur le/les membre(s) depuis un attribut.

Set_Member

Le service Set_Member définit les informations sur le/les membre(s) dans un attribut.

Insert_Member

Le service Insert_Member insère un/des membre(s) dans un attribut.

Remove_Member

Le service Remove_Member supprime un/des membre(s) d'un attribut.

Group_Sync

Le service Group_Sync vérifie que chaque membre d'un groupe est synchronisé à System Time.

6.2.1.2.1.6 Services spécifiques à un objet

Les services spécifiques à un objet sont des services uniques pris en charge seulement par la classe d'objets dans laquelle ils sont définis, tandis que les services communs peuvent être utilisés dans plusieurs objets. Les codes de service spécifiques à un objet doivent être uniques seulement au sein de la classe dans laquelle ils sont définis.

Les services spécifiques à un objet envoyés vers le niveau classe d'un objet sont appelés services (de niveau classe) de gestion de système spécifiques à un objet. Les services spécifiques à un objet envoyés vers le niveau instance d'un objet sont appelés services (de niveau instance) de gestion d'objets spécifiques à un objet.

Pour définir le niveau de classe, il est exigé de spécifier un ID d'instance d'objet de zéro (voir IEC 61158-6-2, 4.1.9.4.1).

6.2.1.2.2 Modèle formel pour Identity

6.2.1.2.2.1 Définition de classe

L'ASE Identity donne des informations d'identification et des informations générales relatives à l'appareil et, éventuellement, à ses sous-systèmes. L'instance 1 de l'objet Identity doit être présente dans tous les appareils.

Instance 1 doit identifier l'ensemble de l'appareil. Elle doit être utilisée seule pour le détrompage électronique, par des applications souhaitant déterminer quels nœuds sont sur le réseau et pour faire correspondre un fichier EDS à un produit sur le réseau.

Les autres instances sont facultatives. Elles peuvent être fournies par un appareil pour donner des informations complémentaires relatives à un appareil, comme ses composants et/ou sous-systèmes intégrés.

Les instances supérieures à 1 qui sont utilisées pour rapporter la révision des composants intégrés doivent se voir attribuer le type d'appareil du composant intégré. Ces instances existent pour permettre au fabricant de rendre compte de la révision des composants intégrés de son choix.

Des instances supérieures à 1 peuvent également être utilisées pour identifier des sous-systèmes internes dont l'exécution est semi-indépendante (par exemple, un appareil peut être conçu pour accepter une carte de communication tierce).

Placé en face d'un attribut ou d'un service, le symbole (*) signifie que cet attribut ou ce service est soit obligatoire, soit facultatif, sur la base de certaines contraintes définies dans la description de l'attribut/du service.

FAL ASE: **FAL Management ASE**
CLASS: **Identity_Object**
CLASS ID: **1**
PARENT CLASS: **Base_Object**
ATTRIBUTS D'ACCES:

1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name

ATTRIBUTS DE GESTION DE SYSTEME (ATTRIBUTS DE CLASSE):

1	(o)	Attribute:	Revision = 1
2	(*)	Attribute:	Max Instance
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes

ATTRIBUTS DE GESTION D'OBJETS (ATTRIBUTS D'INSTANCE):

1	(m)	Attribute:	Vendor ID
2	(m)	Attribute:	Device Type
3	(m)	Attribute:	Product Code
4	(m)	Attribute:	Revision
4.1	(m)	Attribute:	Major Revision
4.2	(m)	Attribute:	Minor Revision
5	(m)	Attribute:	Status
5.1	(m)	Attribute:	Owned
5.2	(m)	Attribute:	Configured
5.3	(m)	Attribute:	Extended Device Status
5.4	(m)	Attribute:	Minor Recoverable Fault
5.5	(m)	Attribute:	Minor Unrecoverable Fault
5.6	(m)	Attribute:	Major Recoverable Fault
5.7	(m)	Attribute:	Major Unrecoverable Fault
5.8	(m)	Attribute:	Extended Device Status 2
6	(m)	Attribute:	Serial Number
7	(m)	Attribute:	Product Name
8	(o)	Attribute:	State
9	(o)	Attribute:	Configuration Consistency Value
10	(o)	Attribute:	Heartbeat Interval
11	(o)	Attribute:	Active Language
12	(o)	Attribute:	Supported Language List
13	(o)	Attribute:	International Product Name
14	(o)	Attribute:	Semaphore
14.1	(m)	Attribute:	Client Electronic Key
14.2	(m)	Attribute:	Semaphore Timer
15	(o)	Attribute:	Assigned Name
16	(o)	Attribute:	Assigned Description
17	(o)	Attribute:	Geographic Location
18	(*)	Attribute	Type15 Identity Info
19	(o)	Attribute	Protection Mode
20	(o)	Attribute	Uptime

SERVICES DE GESTION DE SYSTEME:

1	(o)	Mgt Service:	Get_Attributes_All
5	(o)	Mgt Service:	Reset
14	(*)	Mgt Service:	Get_Attribute_Single
17	(*)	Mgt Service:	Find_Next_Object_Instance
24	(o)	Mgt Service:	Get_Member

SERVICES DE GESTION D'OBJETS:

1	(*)	Ops Service:	Get_Attributes_All
5	(m)	Ops Service:	Reset
14	(m)	Ops Service:	Get_Attribute_Single
16	(*)	Ops Service:	Set_Attribute_Single
24	(*)	Ops Service:	Get_Member

SERVICES SPECIFIQUES A L'OBJET:

75	(o)	Ops Service:	Flash_LEDs

6.2.1.2.2.2 Attributs de gestion de système (attributs de classe)

Max instance

S'il existe plusieurs instances de l'objet Identity, cet attribut est **obligatoire**, sinon il est **facultatif**.

6.2.1.2.2.3 Attributs de gestion d'objets

Les attributs d'instance suivants doivent être utilisés pour identité avec certitude l'appareil approprié visé par un émetteur de connexion:

1) Vendor ID,
2) Device Type,
3) Product Code,
4) Revision.

Cet ensemble d'attributs, lorsqu'ils sont détenus par l'émetteur de connexion, est appelé "clé électronique" d'un appareil.

Vendor ID

Identification de chaque fabricant/vendeur par un numéro. Les Vendor ID identifient de façon univoque un fabricant/vendeur particulier. La valeur zéro ne doit pas être utilisée.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

Device type

Indication de type général de produit. Afin de permettre l'interopérabilité et l'interchangeabilité, un Device Type doit être utilisé pour identifier des appareils semblables qui présentent le même comportement, produisent et/ou consomment le même jeu de données et contiennent le même jeu d'attributs configurables. La définition formelle de ces informations est connue sous le nom de Device Profile (profil d'appareil): tous les appareils avec le même numéro de Device Type doivent satisfaire aux exigences minimales et mettre en œuvre les options communes spécifiées dans le Device Profile pour ce type de données. Une énumération des gammes de Device Type peut être consultée dans l'IEC 61158-6-2. Le type d'appareil du composant intégré n'est pas autorisé pour l'instance 1.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

Product code

Identification d'un produit particulier d'un fabricant individuel. Le Product Code attribué par le fabricant identifie un produit particulier au sein d'un type d'appareil. Chaque fabricant doit attribuer ce code à chacun de ses produits. Le Product Code se met typiquement en correspondance à un ou plusieurs numéros de catalogue/modèle. Les produits doivent avoir des codes différents si les options de configuration et/ou d'exécution sont différentes. De tels appareils présentent une vue logique différente au réseau. La classe zéro n'est pas valide.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

Revision

Révision de l'élément que cette instance de l'objet Identity représente. L'attribut Revision, qui consiste en Major Revision (révision majeure) et Minor Revision (révision mineure), identifie la révision de l'élément que l'objet Identity représente.

La valeur zéro n'est valide pour aucun des champs Major Revision et Minor Revision d'un produit conforme. Si le Device Type de l'instance est un composant intégré, la révision majeure et/ou la révision mineure peuvent être nulles.

NOTE 1 Les Major et Minor Revision sont typiquement affichées sous la forme "major.minor", les révisions mineures étant affichées sous la forme de trois chiffres avec autant de zéros de début que nécessaire.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

Major revision

Il convient que la révision majeure soit incrémentée par le fabricant lorsqu'un changement significatif est apporté à la "disposition, forme ou fonction" du produit. Tout changement qui affecte les choix de configuration disponibles pour l'utilisateur (et par conséquent, le fichier EDS) requiert une incrémentation de la révision majeure.

Minor revision

La révision mineure est typiquement utilisée pour identifier des changements dans un produit qui n'affectent pas les choix de configuration de l'utilisateur, par exemple corrections de bogue, changement de composant matériel, changement d'étiquetage. Les modifications de révision mineure ne sont pas utilisées par un outil de configuration pour faire correspondre un appareil à un fichier EDS.

Status

Statut récapitulatif d'appareil. Cet attribut représente le statut courant de l'appareil entier. Sa valeur varie lorsque l'état de l'appareil change. Le format détaillé de cet attribut Status est décrit dans l'IEC 61158-6-2.

NOTE 2 Les événements qui constituent un défaut (récupérable ou irrécupérable) sont déterminés par les réalisateurs d'appareils.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

Owned

Un (TRUE) indique que l'appareil (ou un objet au sein de l'appareil) a un propriétaire. A l'intérieur du paradigme maître/esclave, la définition de ce bit indique que le jeu de connexions maître/esclave a été alloué à un maître. En dehors du paradigme maître/esclave, la signification de ce bit doit être définie.

Configured

Un (TRUE) indique que l'application de l'appareil a été configurée pour faire quelque chose de différent de la valeur par défaut au déballage ("out–of–box") . Cela ne doit pas inclure la configuration des communications.

Extended device status

Cet attribut est utilisé pour donner des informations plus détaillées relatives au statut de l'appareil. Ses valeurs sont spécifiques au fournisseur ou conformes aux valeurs spécifiées dans l'IEC 61158-6-2, 4.1.8.2.1.2. Le fichier EDS doit indiquer si l'instance d'appareil (1) suit une définition spécifique à un fournisseur pour ces bits. Aucun mécanisme n'est prévu pour énumérer l'utilisation spécifique à un fournisseur de ces valeurs dans le fichier EDS pour des instances supérieures à 1.

Minor recoverable fault

Un (TRUE) indique que l'appareil a détecté un problème en son sein, qui est jugé être récupérable. Le problème ne doit pas amener l'appareil à se mettre dans l'un des états défaillants.

NOTE 3 Par exemple, un appareil d'entrée analogique capte une entrée qui dépasse la valeur maximale configurée de l'entrée.

Minor unrecoverable fault

Un (TRUE) indique que l'appareil a détecté un problème en son sein, qui est jugé être irrécupérable. Le problème ne doit pas amener l'appareil à se mettre dans l'un des états défaillants.

NOTE 4 Par exemple, la RAM soutenue par batterie de l'appareil nécessite un remplacement de batterie. L'appareil est tenu de continuer de fonctionner correctement jusqu'à la première fois que la puissance est mise en cycle.

Major recoverable fault

Un (TRUE) indique que l'appareil a détecté un problème en son sein, qui a amené l'appareil à entrer dans l'état "Major Recoverable Fault" (c'est-à-dire: défaut récupérable majeur).

NOTE 5 Par exemple, la configuration de l'appareil est incorrecte ou incomplète.

Major unrecoverable fault

Un (TRUE) indique que l'appareil a détecté un problème en son sein, qui a amené l'appareil à entrer dans l'état "Major Unrecoverable Fault" (c'est-à-dire: défaut irrécupérable majeur).

Un appareil peut ne pas être capable de communiquer dans l'état "Major Unrecoverable Fault". Par conséquent, il pourrait ne pas être capable de rapporter un Major Unrecoverable Fault (c'est-à-dire: défaut irrécupérable majeur). Il ne doit pas traiter un service Reset. La seule façon de sortir d'un défaut irrécupérable majeur doit être de cycler la puissance de l'appareil.

NOTE 6 Par exemple, l'appareil n'a pas réussi son processus de somme de contrôle de mémoire morte.

Extended device status 2

Cet attribut est utilisé pour donner des informations plus détaillées relatives au statut de l'appareil. Ses valeurs sont spécifiques au vendeur. Le fichier EDS doit indiquer si l'instance d'appareil (1) suit une définition spécifique à un fournisseur pour ces bits. Aucun mécanisme n'est prévu pour énumérer l'utilisation spécifique à un fournisseur de ces valeurs dans le fichier EDS pour des instances supérieures à 1.

Serial number

Numéro de série d'appareil. Cet attribut fournit un nombre utilisé conjointement avec le Vendor ID pour former un identificateur propre à chaque appareil ou tout réseau de type 2. Chaque fabricant a la responsabilité de garantir l'unicité du numéro de série parmi la totalité de ses appareils.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

Product name

Identification lisible par l'homme. Cette chaîne textuelle doit présenter une courte description du produit ou de la famille de produits représenté(e) par l'attribut “Product Code”. Le même code de produit peut avoir une diversité de chaînes de noms de produits.

NOTE 7 La vue logique du réseau (voir la description de Product code ci-dessus) n'inclut pas le Product name.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

State

Indication du présent état de l'appareil, tel que représenté par le diagramme de transitions d'états.

NOTE 8 La nature d'un défaut Major Unrecoverable Fault pourrait faire qu'il ne puisse pas être reflété de façon exacte par l'attribut State.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

Configuration consistency value

Le contenu identifie la configuration de l'appareil. Indique qu'un changement de configuration s'est produit.

Un produit peut modifier automatiquement la Configuration Consistency Value (c'est-à-dire: valeur de cohérence de la configuration) chaque fois qu'un attribut non volatil est altéré. Un nœud client peut, ou peut ne pas, comparer cette valeur à une valeur dans sa propre mémoire avant l'exploitation du système.

NOTE 9 Le comportement du nœud client, à la détection d'une discordance, est spécifique à un vendeur. La Configuration Consistency Value pourrait être un CRC, un compteur d'incrémentation ou tout autre mécanisme.

La seule exigence est que si la configuration change, il convient que la Configuration Consistency Value soit différente afin de refléter le changement.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

Heartbeat interval

Etablit l'intervalle nominal entre productions de messages heartbeat facultatifs.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

Active language

Langue actuellement active pour l'appareil. Si cet attribut est pris en charge, toutes les chaînes de caractères au sein de l'appareil du type de données STRING ou SHORT_STRING doivent suivre ce réglage. Seules les langues prises en charge par le jeu de caractères de l'ISO/IEC 8859-1 peuvent être représentées dans ces chaînes; si l'attribut Active Language est mis à une langue qui ne peut pas être représentée en ISO/IEC 8859-1, l'appareil doit retourner la chaîne dans la langue par défaut. Aucun autre attribut d'objet (niveau classe ou instance) qui établit la langue pour des chaînes de ces types de données ne doit être pris en charge (à savoir l'attribut Native Language de niveau classe des objets Parameter et Parameter Group).

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

Supported language list

Liste des langues prises en charge par les chaînes de caractères du type de données STRINGI au sein de l'appareil

Cet attribut d'instance volatil a une règle d'accès Get.

International product name

Noms du produit dans diverses langues

Cet attribut d'instance non volatil a une règle d'accès Get.

Semaphore

Fournit un sémaphore pour la synchronisation de l'accès client à l'appareil entier.

Il s'agit d'un attribut volatil dont la valeur après réinitialisation ou mise sous tension est toujours zéro. L'attribut Semaphore peut être lu à tout moment. La valeur communiquée sera le contenu courant de l'attribut Semaphore, y compris la valeur de temporisateur (réelle) temps réel.

Client electronic key

Il est composé du Vendor ID du client et du numéro de série de l'appareil.

Semaphore timer

Il s'agit d'un temporisateur en millisecondes qui, lorsqu'il est différent de zéro, compte à rebours jusqu'à zéro. Lorsque le Semaphore Timer atteint la valeur zéro, l'attribut Semaphore est mis à zéro partout.

Le composant Semaphore Timer fonctionne à la résolution temporelle de base de l'appareil. Par conséquent, si la base de temps de l'appareil est supérieure à une résolution de 1, la valeur de temps acceptée par l'attribut doit être arrondie par excès au prochain incrément de temporisateur approprié pour l'appareil.

Lorsqu'un service Set_Attribute_Single est dirigé vers cet attribut, l'attribut sera mis aux données de service si l'une des propositions ci-après est vraie:

- la valeur actuelle contenue dans l'attribut est zéro;
- la partie Electronic Key des données de service concorde avec la partie Electronic Key des données d'attribut.

Cet attribut d'instance a une règle d'accès Get/Set.

Assigned name

Cette chaîne textuelle représente le nom attribué par l'utilisateur de l'appareil.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

Assigned description

Cette chaîne textuelle représente une description attribuée par l'utilisateur de l'appareil et peut aussi décrire sa fonction.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

Geographic location

Cette chaîne textuelle représente l'emplacement physique de l'appareil tel que fourni par l'utilisateur.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

Type15 identity info

Cet attribut est réservé aux appareils de type 15. Il ne doit pas être utilisé dans d'autres cas.

Protection mode

Indication du mode de protection actuel de l'appareil. L'utilisation de cet attribut est décrite plus en détail en 6.2.1.2.2.7.

Cet attribut d'instance volatil a une règle d'accès Get.

Uptime

Durée depuis la mise sous tension ou le redémarrage de l'appareil (par exemple, réinitialisation de l'objet Identity). Le fournisseur détermine où, dans le processus de mise sous tension/redémarrage, cette valeur Uptime commence à être incrémentée, mais cette valeur Uptime ne doit pas commencer à incrémenter plus tard que la transition vers l'état Standby (voir Figure 3).

Cet attribut d'instance volatil a une règle d'accès Get.

6.2.1.2.2.4 Services de gestion de système (niveau classe) et de gestion d'objets (niveau instance)

Les détails spécifiques à un objet relatifs aux services communs sont donnés ci-dessous pour l'objet Identity.

Get_Attributes_All

Ce service est **facultatif** pour CP 2/3 au niveau de l'instance, sinon il est **obligatoire**.

Reset

Le paramètre Object Specific Data de la primitive "request" contiendra un paramètre dédié pour spécifier le type de Reset demandé par l'utilisateur. Son format détaillé et ses options de réinitialisation correspondantes sont spécifiés dans l'IEC 61158-6-2.

Lorsque l'ASE (Objet) Identity reçoit une demande de Reset, il doit:

1) déterminer s'il peut fournir le type de réinitialisation demandé et renvoyer l'erreur appropriée s'il ne peut pas:
2) répondre à la demande;
3) tenter d'exécuter le type de réinitialisation demandé.

Le service Reset ne doit pas être traité lorsque l'appareil est dans l'état "Major Unrecoverable Fault".

L'appareil doit cesser d'accepter les demandes de service de message explicite reçues sur n'importe quel port dès que possible, mais au plus tard 3 s après l'envoi de la réponse de réinitialisation réussie. Les messages qui ne sont pas acceptés peuvent être soit ignorés, soit rejetés par le code de statut de retour 0x10 (conflit d'état de l'appareil). L'appareil doit reprendre l'acceptation de demandes de service de message explicite une fois la réinitialisation terminée.

Il convient qu'un client qui envoie un service Reset à l'appareil attende au moins s après réception d'une réponse d'opération réussie avant de tenter de rétablir les communications avec l'appareil afin de s'assurer que la réponse n'est pas envoyée par l'appareil avant la réinitialisation. Il convient que le client garde à l'esprit que les appareils ne sont pas forcés à effectuer la totalité de la réinitialisation et de la reprise des communications en 3 s, mais qu'ils sont seulement tenus d'arrêter de répondre aux demandes de service avant le début de la réinitialisation réelle en 3 s.

Get_Attribute_Single

Au niveau classe, ce service est **obligatoire** si des attributs sont mis en œuvre, sinon il est **facultatif**.

Set_Attribute_Single

Ce service est **obligatoire** si des attributs réglables sont mis en œuvre (par exemple, Heartbeat Interval), sinon il est **facultatif**.

Find_Next_Object_Instance

Ce service est **obligatoire** s'il existe des instances consécutives au niveau de la classe, sinon il est **facultatif**.

Get_Member

Ce service est **obligatoire** si des attributs de type de données International String (STRINGI) sont mis en œuvre, sinon il est **facultatif**.

6.2.1.2.2.5 Services spécifiques à un objet

Flash_LEDs

Ce service indique à l'appareil de démarrer ou d'arrêter le clignotement de ses DEL définies en Type 2 (utilisées pour l'identifier visuellement).

6.2.1.2.2.6 Diagramme d'états pour Identity

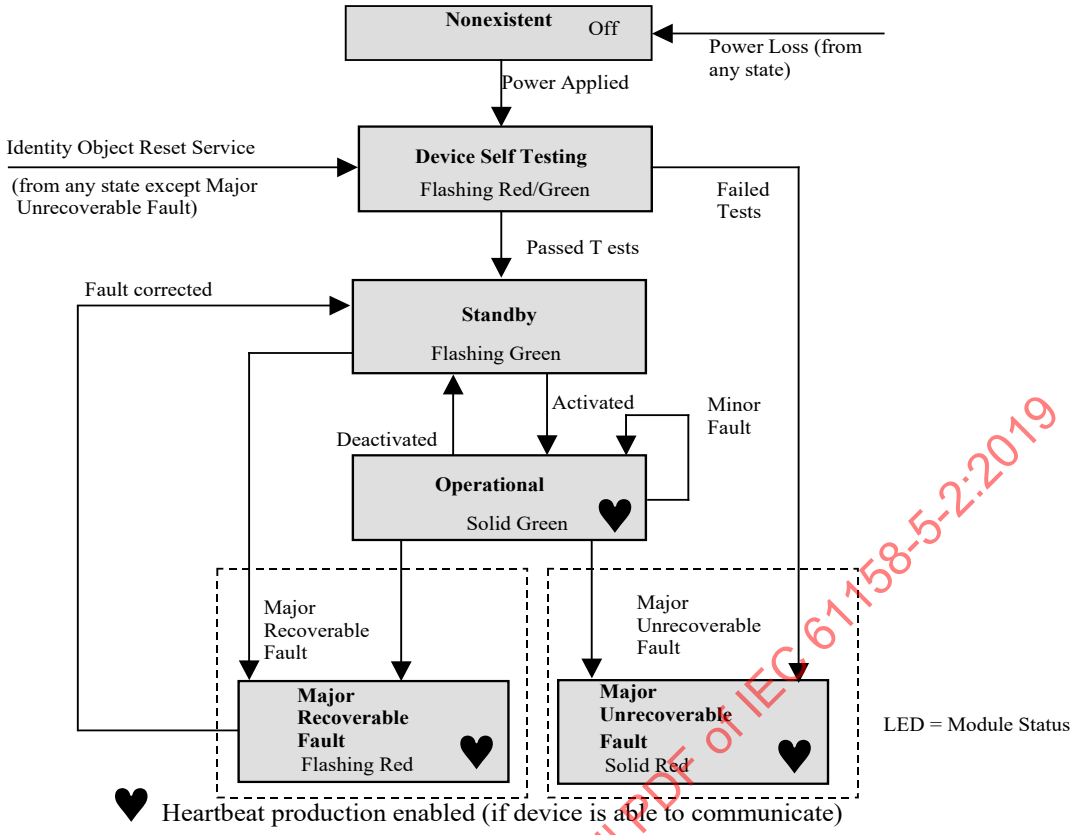
Le comportement de l'objet Identity est montré dans le State Transition Diagram (STD, c'est-à-dire: diagramme de transitions d'états) à la Figure 3. Ce STD associe l'état de l'appareil au statut rapporté par l'attribut Status et à l'état de la DEL Module Status (voir l'IEC 61158-4-2 pour les détails).

Un appareil peut ne pas être capable de communiquer dans l'état "Major Unrecoverable Fault". Par conséquent, il pourrait ne pas être capable de rapporter un Major Unrecoverable Fault (c'est-à-dire: défaut irrécupérable majeur). Il ne traitera pas un service Reset. La seule façon de sortir d'un défaut irrécupérable majeur est de cycler la puissance.

L'objet Identity déclenche la production de messages Heartbeat tels que définis par le réseau sous-jacent lorsque:

- l'intervalle configuré dans l'attribut Heartbeat Interval s'est écoulé depuis le dernier message Heartbeat;
- le contenu du message Heartbeat change, à une fréquence maximale d'un message Heartbeat "donnée changée" par seconde.

La valeur de l'attribut Heartbeat Interval doit être sauvegardée comme attribut non volatil. Les messages Heartbeat sont déclenchés seulement après que l'appareil a parachevé avec succès le diagramme d'états d'accès réseau et est en ligne. Tous les réseaux ne prennent pas en charge l'envoi du message Heartbeat.



Anglais	Français
Nonexistent Off	Nonexistent (Inexistant) Off (Eteint)
Power Loss (from any state)	Perte de puissance (à partir de n'importe quel état)
Power Applied	Puissance appliquée
Identity Object Reset Service (from any state except Major Unrecoverable Fault)	Service de réinitialisation d'objet Identity (à partir de n'importe quel état à l'exception de Unrecoverable Fault)
Device Self Testing Flashing Red/Green	Device Self Testing (Appareil en autotest) Rouge/Vert clignotant
Failed Tests	Echec aux tests
Passed Tests	Réussite des tests
Fault corrected	Défaut corrigé
Standby Flashing Green	Standby (En attente) Vert clignotant
Activated	Activé
Deactivated	Désactivé
Minor Fault	Défaut mineur
Operational Solid Green	Operational (Opérationnel) Vert brillant
Major Recoverable Fault	Défaut récupérable majeur
Major Unrecoverable Fault	Défaut irrécupérable majeur
Major Recoverable Fault Flashing Red	Major Recoverable Fault (Défaut récupérable majeur) Rouge clignotant
Major Unrecoverable Fault Solid Red	Major Unrecoverable Fault (Défaut irrécupérable majeur) Rouge brillant

Anglais	Français
LED = Module Status	DEL = Module Status (Statut du module)
Heartbeat production enabled (if device is able to communicate)	Production de heartbeat activée (si l'appareil est capable de communiquer)

Figure 3 – Diagramme de transitions d'états pour l'objet Identity

Le STD pour l'objet Identity contient les événements suivants:

- Power Applied (Puissance appliquée): l'appareil est mis sous tension;
- Passed Tests (Réussite des essais): l'appareil a passé avec succès tous les autoessais;
- Failed Tests (Echec des essais): l'autoessai de l'appareil a échoué;
- Activated (Activé): la configuration de l'appareil est valide et l'application pour laquelle l'appareil a été conçu est maintenant capable de s'exécuter (des voies de communication peuvent ou peuvent ne pas encore être établies);
- Deactivated (Désactivé): la configuration de l'appareil n'est plus valide et l'application pour laquelle l'appareil a été conçu n'est plus capable de s'exécuter (des voies de communication peuvent ou peuvent ne pas encore être établies);
- Minor fault (Défaut mineur): un défaut classé comme étant soit un défaut irrécupérable mineur, soit un défaut récupérable mineur s'est produit;
- Major recoverable fault (Défaut récupérable majeur): un événement classé comme étant un Major Recoverable Fault s'est produit;
- Major unrecoverable fault (Défaut irrécupérable majeur): un événement classé comme étant un Major Unrecoverable Fault s'est produit;
- Fault Corrected (Défaut corrigé): la source du Major Recoverable Fault a été corrigée.

Le Tableau 7 définit la matrice d'événements d'états pour l'objet Identity.

Tableau 7 – Matrice d'événements d'états pour l'objet Identity

Evénement	Nonexistent (Inexistant)	Device Self Testing (Appareil en autotest)	Standby (En attente)	Operational (Opérationnel)	Major Unrecoverable Fault	Major Recoverable Fault
Power Loss (Perte de puissance)	Non applicable	Transition vers Nonexistent	Transition vers Nonexistent	Transition vers Nonexistent	Transition vers Nonexistent	Transition vers Nonexistent
Puissance appliquée	Transition vers Device Self Testing	Non applicable	Non applicable	Non applicable	Non applicable	Non applicable
Echec aux essais	Non applicable	Transition vers Major Unrecoverable Fault	Non applicable	Non applicable	Non applicable	Non applicable
Réussite des essais	Non applicable	Transition vers Standby	Non applicable	Non applicable	Non applicable	Non applicable
Désactivé	Non applicable	Ignorer l'événement	Ignorer l'événement	Transition vers Standby	Ignorer l'événement	Ignorer l'événement
Activé	Non applicable	Ignorer l'événement	Transition vers Operational	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement
Major Recoverable Fault	Non applicable	Non applicable	Transition vers Major Recoverable Fault	Transition vers Major Recoverable Fault	Ignorer l'événement	Ignorer l'événement
Major Unrecoverable Fault	Non applicable	Non applicable	Transition vers Major Unrecoverable Fault	Transition vers Major Unrecoverable Fault	Ignorer l'événement	Ignorer l'événement
Minor Recoverable Fault	Non applicable	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement
Minor Unrecoverable Fault	Non applicable	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement
Fault Corrected (Défaut corrigé)	Non applicable	Non applicable	Non applicable	Non applicable	Non applicable	Transition vers Standby
Reset	Non applicable	Redémarrer les autotests	Transition vers Device Self Testing	Transition vers Device Self Testing	Ignorer l'événement	Transition vers Device Self Testing
DEL Module Status	Off (Eteint)	Rouge/Vert clignotant	Vert clignotant	Vert brillant	Rouge brillant	Rouge clignotant

Le SEM pour l'objet Identity contient les états suivants:

- Nonexistent (Inexistant): l'appareil est sans énergie;
- Device Self Testing (Appareil en Autotest): l'appareil exécute ses autotests;
- Standby (En attente): l'appareil a besoin d'une mise en service en raison d'une configuration incorrecte ou incomplète;
- Operational (Opérationnel): l'appareil fonctionne d'une façon qui est normale pour l'appareil;
- Major Recoverable Fault (Défaut récupérable majeur): l'appareil a subi un défaut qui est jugé être récupérable;
- Major Unrecoverable Fault (Défaut irrécupérable majeur): l'appareil a subi un défaut qui est jugé être irrécupérable.

6.2.1.2.2.7 Exigences spécifiques pour la manipulation des réglages de la protection

Deux réglages de la protection sont définis: Protection implicite et protection explicite. La valeur par défaut du paramètre implicite et du paramètre explicite n'est pas protégée. Lorsque les paramètres de protection implicite et explicite ne sont pas protégés, l'appareil doit accepter tous les messages explicites de type 2 qui sont valides pour l'appareil, sous réserve des règles spécifiques à un objet ou à un appareil concernant des conflits d'état.

Le paramétrage de la protection implicite doit indiquer que l'appareil est entré dans un état dans lequel il rejette des demandes de messages explicites qui interrompraient son fonctionnement. Les conditions dans lesquelles un appareil entre en mode de protection implicite sont spécifiques à un appareil.

EXEMPLE 1

Quelques exemples de conditions dans lesquelles il convient qu'un appareil prenne en compte l'entrée en mode de protection implicite sont énumérés ci-dessous:

- présence d'une I/O connection (connexion E/S) cible établie. Cette méthode est fortement recommandée pour les appareils dont les principaux moyens de production de commande et/ou de données sont par le biais d'une I/O connection. Pour les connexions O->T qui comprennent l'en-tête Run/Idle, la connexion peut en outre être qualifiée comme étant en mode Run;
- l'appareil est commandé ou fonctionne localement par l'intermédiaire d'une interface utilisateur locale;
- l'appareil est commandé par l'intermédiaire d'une connexion de message explicite.

Les appareils doivent quitter le mode de protection implicite lorsque les conditions dans lesquelles ils ont entrés en mode de protection implicite ne sont plus présentes.

Lorsqu'il est dans le paramètre de protection implicite, l'appareil doit rejeter les messages explicites de type 2 qui seraient disruptifs pour le fonctionnement en cours de l'appareil.

Les éléments suivants énumèrent les demandes de message explicite disruptif qu'un appareil doit rejeter lorsqu'il est en mode de protection implicite:

- service de réinitialisation vers l'objet Identity;
- mise à jour du micrologiciel de l'appareil;
- modifications des paramètres de communication qui provoqueraient la perte de communications avec l'appareil.

Les demandes supplémentaires de message explicite qu'un appareil peut rejeter en mode de protection implicite (Implicit Protection setting) sont propres à un appareil. Les éléments protégés lorsqu'ils sont en mode de protection implicite sont appelés la "politique de protection implicite" (Implicit Protection Policy).

Le paramètre de protection explicite doit indiquer que l'appareil a été explicitement mis dans un mode protégé, soit par des moyens matériels, soit par l'intermédiaire d'un réglage logiciel.

EXEMPLE 2

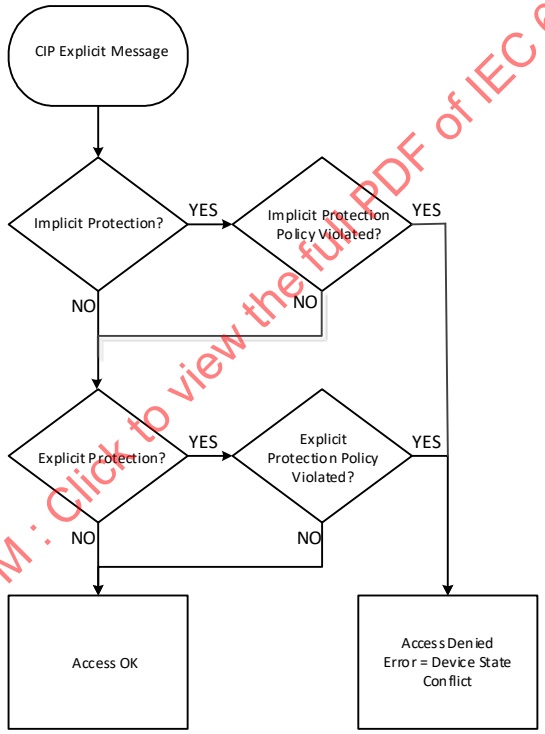
Des exemples de méthodes de mise d'un dispositif dans un mode de protection explicite sont donnés ci-dessous:

- le commutateur à clé de l'appareil est mis sur la position "Run";
- l'interrupteur rotatif du module est réglé sur une valeur spécifique pour activer le réglage de la protection explicite;
- l'appareil est mis en mode de protection explicite par l'intermédiaire d'une interface utilisateur locale (par exemple, écran tactile, clavier, etc.);
- l'appareil est mis en mode de protection explicite par l'intermédiaire d'une interface de serveur web ou d'un autre moyen de connexion sûre.

Lorsqu'il est en mode de protection explicite, l'appareil doit rejeter les demandes de messages explicites de type 2 susceptibles de modifier les paramètres de configuration de l'appareil ou d'être disruptives par rapport au fonctionnement courant de l'appareil. L'appareil peut être équipé d'un mécanisme (spécifique au fournisseur) pour la configuration à partir de laquelle les messages explicites disruptifs doivent être rejetés en mode de protection explicite. Il convient que ce mécanisme soit disponible/accessible uniquement lorsque l'appareil n'est pas protégé.

Un appareil qui met en œuvre l'attribut de mode de protection peut mettre en œuvre le paramètre de protection implicite, le paramètre de protection explicite ou les deux. Lorsque les deux paramètres sont mis en œuvre, il convient que le paramètre de protection explicite inclue par défaut les messages explicites couverts par le paramètre de protection implicite. Les paramètres de protection implicite et explicite peuvent être actifs simultanément, par exemple lorsque l'appareil a été mis en mode de protection explicite, et ont également une I/O connection qui entraîne le rejet des services disruptifs.

La Figure 4 représente le comportement recommandé d'un appareil qui met en œuvre à la fois les paramètres de protection implicite et explicite.



Anglais	Français
CIP Explicit Message	message CIP Explicit
Implicit Protection	Protection implicite
YES	OUI
NO	NON
Implicit Protection Policy Violated	Politique de protection implicite violée
Explicit Protection	Protection explicite
Explicit Protection Policy Violated	Politique de protection explicite violée
Access OK	Accès OK
Access Denied Error = Device State Conflict	Erreur d'accès refusé = Conflit d'état de l'appareil

Figure 4 – Interaction des paramètres explicite et implicite

Les appareils doivent publier dans la documentation utilisateur les conditions dans lesquelles ils entrent dans/sortent de l'un ou l'autre des modes de protection et les éléments couverts par chaque niveau pris en charge. Les éléments protégés lorsqu'ils sont en mode de protection explicite sont appelés la "stratégie de protection explicite" (Explicit Protection Policy).

Tout message explicite rejeté lorsque l'appareil est dans l'un ou l'autre des niveaux de protection doit être retourné avec un code General Status "Device State Conflict" (conflit d'état de l'appareil).

6.2.1.2.3 Modèle formel pour Assembly

6.2.1.2.3.1 Définition de classe

Chaque élément de données (que ce soit une donnée temps réel ou une donnée de configuration) communiqué par un appareil peut être représenté par un attribut d'un des objets internes de l'appareil. La communication de plusieurs éléments de données (attributs) à travers une seule connexion peut exiger que les attributs soient regroupés ou assemblés ensemble en un seul bloc à des fins d'optimisation. Les instances de la classe d'objets Assembly accomplissent ce regroupement. Dans les définitions ci-dessous, les composants individuels sont appelés "membres" de l'assemblage (Assembly).

Un objet Assembly représente un tampon qui lie des attributs de plusieurs objets, permettant que des données à destination ou en provenance de chaque objet soient envoyées ou reçues sur une seule connexion. Des objets Assembly sont utilisés pour produire et/ou consommer des données à destination ou en provenance du réseau. Une instance de l'objet Assembly peut tant produire que consommer des données du réseau s'il est conçu pour ce faire.

Les ASE (objets) Assembly sont soit dynamiques, soit statiques.

- Les assemblages dynamiques utilisent des listes de membres d'assemblages créées et gérées par l'utilisateur. La liste de membres peut être modifiée en ajoutant ou supprimant des membres.
- Les assemblages statiques utilisent des listes de membres définies par le profil d'appareil ou par les réalisateurs de produits. Le numéro d'Instance, le nombre de membres et la liste de membres doivent être fixes.

NOTE Les assemblages statiques peuvent habituellement être entièrement mis en œuvre dans une ROM (mémoire morte).

Les instances de l'objet Assembly sont divisées en plages spécifiées en 4.1.8.4.1.3 de l'IEC 61158-6-2. Des numéros d'instance doivent être affectés aux assemblages dynamiques dans la plage spécifique au vendeur.

Le comportement de l'objet Assembly diffère par le type d'Assembly. Une liste de membres Assembly dynamiques est créée et gérée par l'utilisateur de l'appareil. Le gestionnaire de l'Assembly dynamique doit spécifier et maintenir la liste de membres. Une liste de membres Assembly statiques est définie par le fabricant d'appareils. Elle ne doit pas être modifiée.

Afin de prévoir l'aptitude de créer et de supprimer des objets, ainsi que de modifier des listes de membres, les assemblages dynamiques prennent en charge des services complémentaires qui ne sont pas pris en charge par les assemblages statiques.

Placé en face d'un attribut ou d'un service, le symbole (*) signifie que cet attribut ou ce service est soit obligatoire, soit facultatif, sur la base de certaines contraintes définies dans la description de l'attribut/du service.

FAL ASE:			**FAL Management ASE**
CLASS:			**Assembly_Object**
CLASS ID:			**4**
PARENT CLASS:			**Base_Object**
ATTRIBUTS D'ACCES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
3	(m)	Attribute:	Assembly type (STATIC, DYNAMIC)
ATTRIBUTS DE GESTION DE SYSTEME (ATTRIBUTS DE CLASSE):			
1	(m)	Attribute:	Revision = 2
2	(o)	Attribute:	Max Instance
ATTRIBUTS DE GESTION D'OBJETS (ATTRIBUTS D'INSTANCE):			
1	(*)	Attribute:	Number of Members in List
2	(*)	Attribute:	Member List
2.1	(m)	Attribute:	Member Data Size
2.2	(m)	Attribute:	Member Path Size
2.3	(m)	Attribute:	Member Path
3	(m)	Attribute:	Data
4	(o)	Attribute:	Size
SERVICES DE GESTION DE SYSTEME:			
8	(c)	Constraint:	Assembly type = DYNAMIC
8.1	(m)	Mgt Service:	Create
9	(c)	Constraint:	Assembly type = DYNAMIC
9.1	(o)	Mgt Service:	Delete
14	(*)	Mgt Service:	Get_Attribute_Single
SERVICES DE GESTION D'OBJETS:			
9	(c)	Constraint:	Assembly type = DYNAMIC
9.1	(m)	Ops Service:	Delete
14	(m)	Ops Service:	Get_Attribute_Single
16	(*)	Ops Service:	Set_Attribute_Single
24	(o)	Ops Service:	Get_Member
25	(o)	Ops Service:	Set_Member
26	(c)	Constraint:	Assembly type = DYNAMIC
26.1	(*)	Ops Service:	Insert_Member
27	(c)	Constraint:	Assembly type = DYNAMIC
27.1	(*)	Ops Service:	Remove_Member

6.2.1.2.3.2 Attributs de gestion de système (attributs de classe)

Aucune exigence spécifique pour cet objet.

6.2.1.2.3.3 Attributs de gestion d'objets

Les règles d'accès (Get/Set) pour les attributs d'instance sont en plus limitées par l'état de fonctionnement de l'objet Assembly. Ces contraintes supplémentaires sont détaillées en 6.2.1.2.3.5.

Number of members in list

Cet attribut d'instance est **facultatif** pour un Assembly statique, **obligatoire** pour un Assembly dynamique.

Nombre de membres dans l'attribut "Member List" en dessous.

Cet attribut d'instance a une règle d'accès Get seulement.

Member list

Cet attribut d'instance est **facultatif** pour un Assembly statique, **obligatoire** pour un Assembly dynamique.

La liste de membre est une matrice de taille et origine (chemin interne) de chaque membre. Chaque membre dans la Member List contient les entrées définies ci-dessous.

Cet attribut d'instance a une règle d'accès Get seulement pour un Assembly statique, Set pour un Assembly dynamique.

Member data size

Taille des données de membre (en bits).

Member path size

Taille de Member Path (chemin de membre, en octets). Si aucun chemin n'est spécifié, une valeur de zéro doit être utilisée.

Member path

Chemin interne à destination/en provenance de cet élément (format condensé). Voir l'IEC 61158-6-2 pour le format de chemin condensé.

Les règles suivantes s'appliquent à l'attribut "Member List".

Lorsqu'un chemin vide (Member Path Size = 0) est utilisé dans une liste de membres Assembly, l'Assembly doit insérer/rejeter le nombre de bits tel que spécifié dans le champ du sous-attribut Member Data Size lors de la production/consommation. L'objet Assembly doit insérer s'il produit et rejette s'il consomme. L'Assembly doit utiliser une valeur de zéro pour toutes les données produites qui ont été insérées.

NOTE 1 L'utilisation d'un chemin consommé vide permet, par exemple, que des données destinées à plusieurs nœuds soient envoyées dans un même message, car chaque nœud peut être configuré pour rejeter les données dans le message qui ne lui sont pas destinées.

Le chemin vide doit être pris en charge pour tous les assemblages. Les assemblages statiques peuvent également contenir des chemins vides.

Aucune vérification n'est requise de l'objet Assembly à aucun moment pour s'assurer que la taille des données de membre est correcte pour le chemin de membre donné. Il est de la responsabilité du membre Assembly de gérer correctement le trop grand ou le trop faible nombre de données. L'Assembly doit délivrer le nombre configuré de bits au membre.

Aucun bourrage ne doit être effectué par l'objet Assembly lui-même pour aligner les données issues de chaque membre Assembly sur une frontière d'octet, de mot ou autre frontière. Des chemins vides peuvent être utilisés pour fournir le bourrage si tel est le souhait.

Data

La totalité des données du membre condensées en une seule matrice.

NOTE 2 Ces données peuvent contenir plusieurs types de données différents. Pour l'efficacité, il vaut mieux maintenir ce mot de données aligné en le condensant sur les frontières des mots et en ajoutant un bourrage selon les besoins. Ceci peut être accompli en utilisant des "chemins vides" (Member Path Size = 0)

Cet attribut d'instance a une règle d'accès Set.

Size

Nombre d'octets dans l'attribut "Data".

Cet attribut d'instance a une règle d'accès Get seulement.

6.2.1.2.3.4 Services de gestion de système (niveau classe) et de gestion d'objets (niveau instance)

Delete

Au niveau instance (gestion d'objets), le service Delete supprime l'instance d'objet Assembly spécifiée et libère toutes les ressources associées. Au niveau classe (gestion de système), ce service supprime toutes les instances d'Assembly existantes.

Get_Attribute_Single

Ce service est **obligatoire** si des attributs de classe ou l'attribut d'instance Member List sont implémentés au niveau de la classe, sinon il est **facultatif**.

Set_Attribute_Single

Le service Set_Attribute_Single modifie le contenu de l'attribut spécifié de l'instance d'objet Assembly spécifiée. Ce service est **facultatif** pour un Assembly statique et **conditionnel** pour un Assembly dynamique. Ce service peut être utilisé pour ajouter ou retirer un élément dans l'Assembly Member List d'un Assembly dynamique. Si ce service n'est pas pris en charge pour un Assembly dynamique, les services Insert_Member et Remove_Member doivent être pris en charge.

Get_Member

Le service Get_Member renvoie un membre depuis les attributs d'instance Member List et Data.

Set_Member

Le service Set_Member modifie un membre des attributs d'instance Member List et Data.

Insert_Member, Remove_Member

Le service Insert_Member ajoute un membre à l'attribut Member List d'un Assembly dynamique. Le service Remove_Member supprime un membre de l'attribut Member List d'un Assembly dynamique.

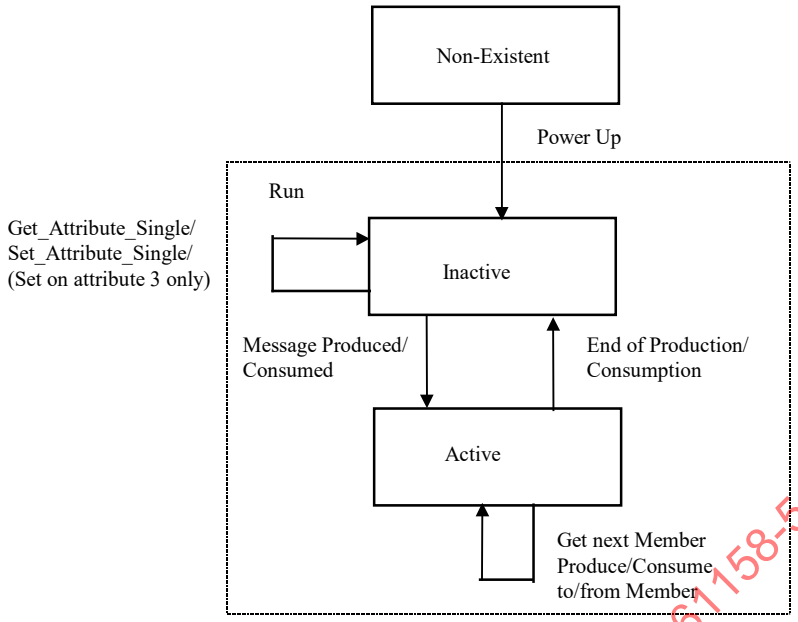
Si ces services ne sont pas pris en charge pour un Assembly dynamique, le service Set_Attribute_Single doit être pris en charge.

6.2.1.2.3.5 Diagrammes d'états pour Assembly

L'utilisation réelle des services pris en charge (par exemple, Get_Attribute_Single et Set_Attribute_Single) est limitée par l'état de fonctionnement de l'objet Assembly. Ces contraintes sont détaillées dans les diagrammes d'états pour Assembly ci-dessous.

Diagramme d'états pour Assembly statique

La Figure 5, le Tableau 8 et le Tableau 9 illustrent le comportement des objets Assembly statiques.



Anglais	Français
Non-Existent	Non-existent (Inexistant)
Power Up	Mise sous tension
Get_Attribute_Single/Set_Attribute_Single/(Set on attribute 3 only)	Get_Attribute_Single/Set_Attribute_Single/(Mis sur attribut 3 seulement)
Run	Marche
Inactive	Inactive (Inactif)
Message Produced/Consumed	Message produit/consommé
End of Production/Consumption	Fin de production/Consommation
Active	Active (Actif)
Get next Member	Récupérer le prochain membre
Produce/Consume to/from Member	Produire/Consommer dans le membre

Figure 5 – Diagramme de transitions d'états pour Assembly statique

Tableau 8 – Matrice d'événements d'états pour Assembly statique

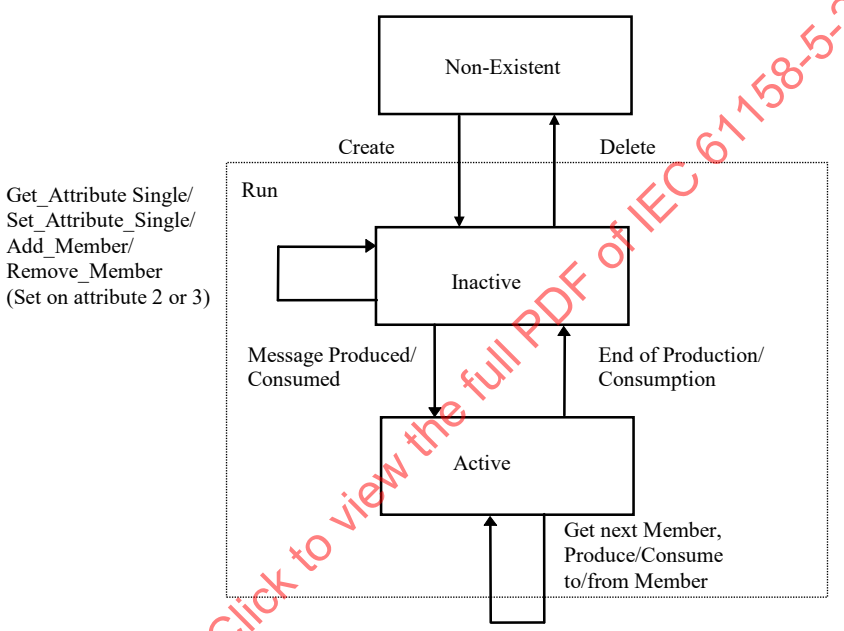
Evénement	Etat de l'objet Assembly statique		
	Non-existent	Inactive (Inactif)	Active (Actif)
Mise sous tension	Transition vers Inactive	Non applicable	Non applicable
Get_Attribute_Single	Erreur: L'objet n'existe pas.	Valider/gérer la demande. Retourner la réponse	Valider/gérer la demande. Retourner la réponse
Set_Attribute_Single	Erreur: L'objet n'existe pas.	Valider/gérer la demande. Retourner la réponse	Erreur: Object state conflict (Conflit d'état d'objet)
Message produced/ consumed (Message produit/consommé)	Erreur: L'objet n'existe pas.	Commencer à produire/consommer dans chaque membre de la liste Transition vers Active	Erreur: Object state conflict (Conflit d'état d'objet)
End of production/ consumption (Fin de production/consommation)	Erreur: L'objet n'existe pas.	Erreur: Object state conflict (Conflit d'état d'objet)	Transition vers Inactive

Tableau 9 – Accès pour les attributs d'instance Assembly statique

Attribut	Etat de l'objet Assembly statique		
	Non-existent	Inactive (Inactif)	Active (Actif)
Number of Members in List	Non disponible	Read Only (Lecture seule)	Read Only (Lecture seule)
Member List	Non disponible	Read Only (Lecture seule)	Read Only (Lecture seule)
Data	Non disponible	Read/Write (Lecture/Ecriture)	Read Only (Lecture seule)

Diagramme d'états pour Assembly dynamique

La Figure 6 et le Tableau 10 illustrent le comportement des objets Assembly dynamiques.



Anglais	Français
Non-Existent	Non-existent (Inexistant)
Create	Créer
Delete	Supprimer
Get_Attribute Single/Set_Attribute_Single/Create/Delete	Get_Attribute Single/Set_Attribute_Single/Create/Delete
Run	Marche
Inactive	Inactive (Inactif)
Message Produced/Consumed	Message produit/consommé
End of Production/Consumption	Fin de production/Consommation
Active	Active (Actif)
Get next Member,	Récupérer le prochain membre
Produce/Consume to/from Member	Produire/Consommer dans le membre

Figure 6 – Diagramme de transitions d'états pour Assembly dynamique

Tableau 10 – Matrice d'événements d'états pour Assembly dynamique

Evénement	Etat de l'objet Assembly dynamique		
	Non-existent	Inactive (Inactif)	Active (Actif)
Create	La classe instancie un objet Assembly. Transition vers Inactive	Non applicable	Non applicable
Delete	Erreur: L'objet n'existe pas.	Libérer toutes les ressources associées. Transition vers Non-existent	Erreur: Object state conflict (Conflit d'état d'objet)
Get_Attribute_Single	Erreur: L'objet n'existe pas.	Valider/gérer la demande. Retourner la réponse	Valider/gérer la demande. Retourner la réponse
Set_Attribute_Single	Erreur: L'objet n'existe pas.	Valider/gérer la demande. Retourner la réponse	Erreur: Object state conflict (Conflit d'état d'objet)
Insert_Member	Erreur: L'objet n'existe pas.	Valider/gérer la demande. Retourner la réponse	Erreur: Object state conflict (Conflit d'état d'objet)
Remove_Member	Erreur: L'objet n'existe pas.	Valider/gérer la demande. Retourner la réponse	Erreur: Object state conflict (Conflit d'état d'objet)
Message produced/ consumed (Message produit/consommé)	Erreur: L'objet n'existe pas.	Commencer à produire/consommer dans chaque membre de la liste Transition vers Active	Erreur: Object state conflict (Conflit d'état d'objet)
End of production/ consumption (Fin de production/consommation)	Erreur: L'objet n'existe pas.	Erreur: Object state conflict (Conflit d'état d'objet)	Transition vers Inactive

Pour tous les attributs d'un objet Assembly dynamique, les services Get_Attribute_Single et Set_Attribute_ Single (lorsqu'ils sont pris en charge) sont seulement disponibles dans l'état Inactive.

Tableau 11 – Accès pour les attributs d'instance Assembly dynamique

Attribut	Etat de l'objet Assembly dynamique		
	Non-existent	Inactive (Inactif)	Active (Actif)
Number of Members in List	Non disponible	Read Only (Lecture seule)	Read Only (Lecture seule)
Member List[a]	Non disponible	Read/Write (Lecture/Ecriture)	Read Only (Lecture seule)
Data	Non disponible	Read/Write (Lecture/Ecriture)	Read Only (Lecture seule)

[a] Cet attribut peut être défini par le service Insert_Member (un membre à la fois) ou par le service Set_Attribute_Single (tous les membres à la fois).

6.2.1.2.3.6 Exigences spécifiques pour une connexion à l'objet Assembly

L'objet Assembly doit être unique parmi tous les objets en ce que ses points de connexion et ses instances sont les mêmes. Par exemple, le point de connexion 4 de l'objet Assembly doit être le même que l'instance 4 de l'objet Assembly. L'accès à des données avec un chemin spécifiant "classe=Assembly, instance=VV, attribut=3" doit retourner les mêmes données qu'un chemin spécifiant "classe= Assembly, point de connexion=VV, attribut=3".

Lorsque l'objet Assembly est la cible d'une connexion, une instance plus deux points de connexion doivent être spécifiés par le chemin de connexion dans le service Forward_Open.

Le format du chemin de connexion doit être "classe=Assembly, instance=XX, point de connexion=YY, point de connexion=ZZ" où XX est l'instance, et YY/ZZ sont les points de connexion. Tout segment de donnée dans le chemin doit être utilisé pour établir l'attribut de données d'instance XX. Le point de connexion (instance) YY doit être le consommateur (O⇒T) pour la connexion. De même, le point de connexion (instance) ZZ doit être le producteur (T⇒O).

En utilisant ce format, trois instances doivent être spécifiées pour une connexion à l'objet Assembly:

- configuration,
- producteur,
- consommateur.

Par exemple, les deux chemins de connexion suivants spécifient la même instance de producteur 8. Ces deux connexions peuvent être faites simultanément, à condition que la connexion demandée spécifie multipoint.

- classe=Assembly, instance=5, point de connexion=4, point de connexion=8
- classe=Assembly, instance=3, point de connexion=2, point de connexion=8

6.2.1.2.4 Modèle formel pour Message Router

6.2.1.2.4.1 Définition de classe

L'ASE (objet) Message Router fournit un point formel de connexion de messagerie par lequel un client peut adresser un service à n'importe quelle classe ou instance d'objet résidant dans l'appareil physique. Chaque nœud doit contenir l'instance 1 de l'objet Message Router.

L'ASE (objet) Message Router doit recevoir tous les messages entrants (indications de service adressées à des objets application ou utilisateur au sein de l'appareil). La source de ces messages peut être soit le gestionnaire de messages non connectés (UCMM), soit une connexion active à l'objet Message Router.

L'objet Message Router doit analyser la première partie du message (informations de chemin) pour déterminer la classe d'objets cible.

Il doit commencer par traiter le segment Electronic Key et les segments Network (si présents dans le chemin). Une erreur dans le segment Electronic Key doit provoquer l'envoi d'une réponse d'erreur avec le statut "Key Failure in path".

L'interprétation du chemin d'application (par exemple, instance de classe) doit ensuite être effectuée sur chaque service reçu par le Message Router. Si le chemin d'application peut être interprété, le service doit être acheminé en interne vers l'objet cible. Tout chemin d'application qui ne peut être interprété par la mise en œuvre d'un Message Router dans un appareil doit faire retourner une réponse d'erreur avec le statut "Object Not Found".

L'indication de service doit ensuite être acheminée vers l'objet-cible en vue d'un traitement réel. Si le service est dirigé vers le Message Router lui-même (services de gestion de système et d'objet adressant le Message Router), le Message Router doit traiter la demande de service et répondre en conséquence.

Lorsque la réponse de service a été reçue de l'objet-cible (ou générée par le Message Router lui-même), elle doit être renvoyée à la source de la demande de service en suivant le chemin inverse. Toutes les réponses de service connecté doivent être retournées à travers la connexion en provenance de laquelle la demande de service a été reçue. Toutes les réponses de service non connecté (UCMM) doivent être retournées par l'intermédiaire de la même transaction UCMM qui a fourni la demande.

NOTE Dans la pratique, le Message Router agit comme un tableau de commutation interne, qui retient toutes les nécessaires informations d'acheminement pour les messages. Par conséquent, les objets-cibles peuvent ne pas connaître ces informations (l'appariement des indications et des réponses de service est obtenu par le biais d'un Local ID fourni par le Message Router).

FAL ASE:			**FAL Management ASE**
CLASS:			**Message_Router_Object**
CLASS ID:			**2**
PARENT CLASS:			**Base_Object**
ATTRIBUTS D'ACCES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
ATTRIBUTS DE GESTION DE SYSTEME (ATTRIBUTS DE CLASSE):			
1	(o)	Attribute:	Revision = 1
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
ATTRIBUTS DE GESTION D'OBJETS (ATTRIBUTS D'INSTANCE):			
1	(o)	Attribute:	Object_List
1.1	(m)	Attribute:	Number
1.2	(m)	Attribute:	Classes
2	(o)	Attribute:	Number available
SERVICES DE GESTION DE SYSTEME:			
1	(o)	Mgt Service:	Get_Attributes_All
14	(*)	Mgt Service:	Get_Attribute_Single
SERVICES DE GESTION D'OBJETS:			
1	(o)	Ops Service:	Get_Attributes_All
10	(o)	Ops Service:	Multiple_Service_Packet
14	(*)	Ops Service:	Get_Attribute_Single
SERVICES SPECIFIQUES A L'OBJET:			
75	(o)	Mgt Service:	Symbolic_Translation

6.2.1.2.4.2 Attributs de gestion de système (attributs de classe)

Aucune exigence spécifique pour cet objet.

6.2.1.2.4.3 Attributs de gestion d'objets

Object_List

Liste de codes de classe d'objets pris en charge par l'appareil par le biais du Message Router.

Cet attribut d'instance a une règle d'accès Get seulement.

Number

Nombre de classes prises en charge dans l'attribut Classes en dessous.

Classes

Liste de codes de classe pris en charge par l'appareil

Number available

Nombre maximal de connexions prises en charge par le Message Router

Cet attribut d'instance a une règle d'accès Get seulement.

6.2.1.2.4.4 Services de gestion de système (niveau classe) et de gestion d'objets (niveau instance)

Get_Attribute_Single

Ce service est **obligatoire** si des attributs sont mis en œuvre, sinon il est **facultatif**.

6.2.1.2.4.5 Services spécifiques à un objet

Symbolic_Translation

Ce service fournit une traduction depuis un chiffrement de chemin Symbolic Segment vers un chiffrement de chemin Logical Segment équivalent, le cas échéant.

Ce service fournit à un appareil dans lequel des représentations Symbolic Segment des adresses de chemin interne sont mises en œuvre, un mécanisme permettant de traduire ces chiffrements d'adresses symboliques en des chiffrements de chemin Logical Segment équivalents.

6.2.1.2.4.6 Exigences spécifiques de l'objet Message Router

L'objet Message Router doit avoir un état, l'état Run. Il doit toujours être en marche (après mise sous tension) et doit être fixé par conception de l'appareil. Plus d'informations relatives à l'objet Message Router peuvent être consultées dans l'IEC 61158-6-2 (définition de Path pour accéder à l'élément d'objet au sein de l'appareil).

L'objet Message Router doit prendre en charge une ou plusieurs connexions à ceux-ci. Un service Forward_Open avec les paramètres énumérés dans le Tableau 12 doit aussi être pris en charge.

Tableau 12 – Paramètres de Forward_Open de l'objet Message Router

Paramètre	Valeur de paramètre
O⇒T priority	faible
O⇒T connection type	point à point
O⇒T fixed/variable	variable
O⇒T size	jusqu'à 504 octets
T⇒O priority	faible
T⇒O connection type	point à point
T⇒O fixed/variable	variable
T⇒O size	jusqu'à 504 octets
Client/server	serveur
Trigger mode	ignorer
Transport class	classe 3 (vérifiée)

6.2.1.2.5 Modèle formel pour Acknowledge Handler

6.2.1.2.5.1 Définition de classe

L'objet Acknowledge Handler est utilisé pour gérer la réception d'acquittements de messages. Cet objet communique avec un objet d'application productrice de messages au sein d'un appareil. L'objet Acknowledge Handler notifie à l'application productrice la réception de l'acquittement; les temporisations d'acquittement, et la limite des répétitions de tentative de production.

FAL ASE:			**FAL Management ASE**
CLASS:			**AckHandler_Object**
CLASS ID:			**43**
PARENT CLASS:			**Base_Object**
ATTRIBUTS D'ACCES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
ATTRIBUTS DE GESTION DE SYSTEME (ATTRIBUTS DE CLASSE):			
1	(o)	Attribute:	Revision = 1
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
ATTRIBUTS DE GESTION D'OBJETS (ATTRIBUTS D'INSTANCE):			
1	(m)	Attribute:	Acknowledge Timer
2	(m)	Attribute:	Retry Limit
3	(m)	Attribute:	COS Producing Connection Instance
4	(o)	Attribute:	Ack List Size
5	(o)	Attribute:	Ack List
6	(o)	Attribute:	Data with Ack Path List Size
7	(o)	Attribute:	Data with Ack Path List
SERVICES DE GESTION DE SYSTEME:			
8	(o)	Mgt Service:	Create
9	(o)	Mgt Service:	Delete
14	(o)	Mgt Service:	Get_Attribute_Single
SERVICES DE GESTION D'OBJETS:			
9	(o)	Ops Service:	Delete
14	(m)	Ops Service:	Get_Attribute_Single
16	(m)	Ops Service:	Set_Attribute_Single
SERVICES SPECIFIQUES A L'OBJET:			
75	(o)	Ops Service:	Add_AckData_Path
76	(o)	Ops Service:	Remove_AckData_Path

6.2.1.2.5.2 Attributs de gestion de système (attributs de classe)

Aucune exigence spécifique pour cet objet.

6.2.1.2.5.3 Attributs de gestion d'objets

Acknowledge timer

Temps à attendre un acquittement avant de renvoyer.

Si la valeur spécifiée pour l'attribut Acknowledge Timer n'est pas égale à un incrément de la résolution d'horloge disponible, la valeur est arrondie par excès à la valeur pratique suivante.

EXEMPLE Une demande de Set_Attribute_Single est reçue et spécifie la valeur 5 pour l'attribut Acknowledge Timer et le produit assure une résolution de 10 ms sur les temporisateurs. Dans ce cas, le produit chargerait la valeur 10 dans l'attribut Acknowledge Timer.

La valeur qui est effectivement chargée dans l'attribut Acknowledge Timer est rapportée dans le Service Data Field (champ de données de service) d'un message de réponse Set_Attribute_Single associé à une demande de modifier cet attribut.

Cet attribut d'instance a une règle d'accès Get/Set.

Retry limit

Nombre d'Ack Timeouts (temporisations d'acquittements) à attendre avant d'informer l'application productrice d'un événement RetryLimit_Reached. Un Set_Attribute_Single réussi à l'attribut Retry Limit réinitialisera le Retry Counter (compteur des répétitions de tentative).

Cet attribut d'instance a une règle d'accès Get/Set (Set facultatif).

COS producing connection instance

Instance de connexion qui contient le chemin de l'objet d'application E/S productrice auquel sera notifié des événements Ack Handler.

Cet attribut d'instance a une règle d'accès Get seulement lorsque l'instance d'objet Acknowledge Handler est active (au moins un membre dans l'Ack List). Il a une règle d'accès de Get/Set lorsque l'instance d'objet Acknowledge Handler est inactive.

Ack list size

Nombre maximal de membres dans l'Ack List.

Cet attribut d'instance a une règle d'accès Get seulement.

Ack list

Liste d'instances de connexion actives qui reçoivent des acquittements. L'attribut Ack List est mis à jour lorsqu'une connexion associée effectue une transition entre "configuring", "established", "timed-out", et "non-existent". Pour les détails, voir la matrice d'événements d'états en 6.2.1.2.5.6.

Cet attribut d'instance a une règle d'accès Get seulement.

Data with ack path list size

Nombre maximal de membres dans Data avec Ack Path List.

Cet attribut d'instance a une règle d'accès Get seulement.

Data with ack path list

Liste de paires instance de connexion/objet d'application consommatrice. Cet attribut est utilisé pour transmettre des données reçues avec acquittement.

Cet attribut d'instance a une règle d'accès Get seulement.

6.2.1.2.5.4 Services de gestion de système (niveau classe) et de gestion d'objets (niveau instance)

Create

Au niveau instance (gestion d'objets), le service Create crée un objet Acknowledge Handler.

Delete

Au niveau instance (gestion d'objets), le service Delete supprime l'objet Acknowledge Handler spécifié. Au niveau classe (gestion de système), ce service supprime toutes les instances d'Acknowledge Handler dynamiquement créées.

6.2.1.2.5.5 Services spécifiques à un objet

Add_AckData_Path

Ce service ajoute un chemin pour les données avec acquittement pour un consommateur connecté.

Remove_AckData_Path

Ce service retire un chemin pour les données avec acquittement pour le consommateur connecté donné.

6.2.1.2.5.6 Diagrammes d'états pour Acknowledge Handler

Le Tableau 13 est la matrice d'événements d'états pour l'objet Acknowledge Handler associé à une connexion Change of State (changement d'état (COS)). Le Tableau 14 est la matrice d'événements d'états pour l'objet d'application E/S productrice.

Tableau 13 – Matrice d'événements d'états de l'objet Acknowledge Handler

Evénement	Etat de l'objet Acknowledge Handler		
	Non-existent	Inactive (Inactif)	Active (Actif)
Receive Acknowledge (Recevoir l'acquittement)	Non applicable	Ignorer l'événement	1) Effacer le fanion Ack. 2) Transmettre toutes les données éventuelles à l'objet d'application. SI tous les acquittements reçus: 3a) Effacer Ack Timer et Retry Counter. 3b) Envoyer l'événement Acknowledge_Received. 3c) Message à l'objet d'application productrice.
Acknowledge Timer expires (Temporisateur d'acquittements expire)	Non applicable	Ignorer l'événement	Envoyer le message d'événement Ack_Timeout à l'objet d'application productrice.
Data sent (Données envoyées)	Non applicable	Ignorer l'événement	1) Mettre le fanion Ack Required. 2) Mettre Ack Timer. 3) Effacer Retry Counter.
Data resent (Données envoyées de nouveau)	Non applicable	Ignorer l'événement	1) Mettre le fanion Ack Required & Ack Timer. SI Retry_Limit > 0 2a) Incrémenter Retry Counter. 2b) SI Retry Counter = Retry_Limit. Envoyer le message événement Retry_Limit_Reached à l'objet d'application productrice.
Delete	Non applicable	Transition vers Non-existent	Transition vers Non-existent
Create	Transition vers Inactive	Non applicable	Non applicable
Apply attributes (Appliquer les attributs)	Non applicable	Vérifier qu'une nouvelle connexion peut être ajoutée à la liste. Transmettre ce message à l'objet d'application consommée, si l'un est configuré pour cette connexion.	Vérifier qu'une nouvelle connexion peut être ajoutée à la liste. Transmettre ce message à l'objet d'application consommée, si l'un est configuré pour cette connexion.
Connection transition to Established (Transition de la connexion vers Established)	Non applicable	Ajouter cette instance de connexion à la liste de connexion (ou signaler en interne comme étant "Acking" (en processus d'acquittement)). Transmettre ce message à l'objet d'application consommée, si l'un est configuré pour cette connexion. Envoyer le message d'événement Acknowledge_Active à l'application productrice. Transition vers Active	Ajouter cette instance de connexion à la liste de connexions. Transmettre ce message à l'objet d'application consommée, si l'un est configuré pour cette connexion.

Evénement	Etat de l'objet Acknowledge Handler		
	Non-existent	Inactive (Inactif)	Active (Actif)
Inactivity/Watchdog Timer expires (Le temporisateur Inactivity/Watchdog Timer expire)	Non applicable	Non applicable	1) Signaler en interne cette connexion comme étant "Not Acking" (non en processus d'acquittement). Un acquittement ne sera plus surveillé pour cette connexion, mais elle reste toutefois dans la liste des connexions. 2) Transmettre cet événement à l'objet d'application consommée, si l'un est configuré pour cette connexion. 3) Si aucune connexion "Acking" dans la liste, envoyer l'événement Acknowledge_Inactive à l'application productrice et passer à Inactive.
Connection deleted (Connexion supprimée)	Non applicable	Ignorer l'événement	1) Retirer cette instance de connexion de la liste des connexions. 2) Transmettre cet événement à l'objet d'application consommée, si l'un est configuré pour cette connexion. 3) Si aucune connexion "Acking" dans la liste, envoyer l'événement Acknowledge_Inactive à l'application productrice et passer à Inactive.

Tableau 14 – Matrice d'événements d'états de l'objet d'application E/S productrice

Evénement	Etat de l'objet d'application E/S productrice			
	Not running	Running	Running with acknowledgment	Prohibited
Change of state detected (Changement d'état détecté)	Ignorer l'événement	1) Informer le Link Producer (producteur de liaison) d'envoyer des données. SI Inhibit Time (Temps de neutralisation) configuré: 2a) Démarrer Inhibit Timer. 2b) Transition vers Produced.	1) Informer le Link Producer (producteur de liaison) d'envoyer des données. 2) Envoyer le message d'événement DataSent à l'objet Acknowledge Handler.. SI Inhibit Time configuré: 3a) Démarrer Inhibit Timer. 3b) Transition vers Prohibited.	Placer l'événement en file d'attente.
Acknowledge received (Acquittement reçu)	Non applicable	Non applicable	Ignorer l'événement	SI Ack_Active mis 1a) SI Inhibit Timer pas en marche. Transition vers "Running with Acknowledgement". 1b) SINON Mettre le fanion Ack_Receive SINON ignorer l'événement.
Acknowledge timeout (Temporisation d'acquittement)	Ignorer l'événement	Non applicable	1) Informer le Link Producer d'envoyer des données. 2) Envoyer le message d'événement Data_Resent à l'objet Acknowledge Handler.	1) Informer le Link Producer d'envoyer des données. 2) Envoyer le message d'événement Data_Resent à l'objet Acknowledge Handler.

Evénement	Etat de l'objet d'application E/S productrice			
	Not running	**Running**	**Running with acknowledgment**	**Prohibited**
Transmission timer expires (Le temporisateur d'émission expire)	Non applicable	Informer le Link Producer d'envoyer des données.	1) Informer le Link Producer d'envoyer des données. 2) Envoyer le message d'événement Data_Sent à l'objet Acknowledge Handler.	1) Informer le Link Producer d'envoyer des données LAST. 2) Envoyer le message d'événement Data_Sent à l'objet Acknowledge Handler.
Retry limit reached (Limite atteinte pour les répétitions de tentative)	Ignorer l'événement	Non applicable	Spécifique au produit	Transition vers Running with Acknowledgement
Inhibit timer expires (Temporisateur de neutralisation expire)	Ignorer l'événement	Non applicable	Non applicable	SI Ack_Active mis 1a) SI Ack_Received Transition vers Running w/Ack. Effacer le fanion Ack_Received. 1b) SINON Ignorer l'événement. SINON Transition vers Running.
Connection deleted (Connexion supprimée)	Non applicable	Transition vers Not Running	Transition vers Not Running	Transition vers Not Running
Acknowledge active (Acquittement actif)	Mettre le fanion Ack Active.	1) Transition vers Running with Acknowledgement. 2) Mettre le fanion Ack Active.	Ignorer l'événement	Mettre le fanion Ack_Active.
Acknowledge Inactive (Acquittement inactif)	Réinitialiser le fanion Ack Active.	Ignorer l'événement	1) Transition vers Running. 2) Réinitialiser le fanion Ack Active.	Réinitialiser le fanion Ack_Active.
Connection transition to Established (Transition de la connexion vers Established)	SI Ack Active Transition vers Running with Acknowledgment. SI Ack Inactive Transition vers Running.	Ignorer l'événement	Ignorer l'événement	Ignorer l'événement
NOTE Il s'agit d'une matrice partielle d'événements d'états pour l'objet Producing I/O Application (application E/S productrice). Seuls les états et événements associés avec acquittement de données sont définis. Les autres états et événements seront plus vraisemblablement associés à un objet d'application E/S productrice.				

6.2.1.2.5.7 Exigences spécifiques pour le comportement et la configuration

Comportement et configuration de la production de données acquittées

Les règles suivantes sont utilisées pour configurer et déterminer le comportement d'une connexion Change of State ou Cyclic I/O acquittée en utilisant l'objet Acknowledge Handler. Dans les exemples suivants, le Producteur de COS ("COS Producer", Producteur de changement d'état) est utilisé pour référencer l'appareil produisant des données de changement d'état ou cycliques et consommant un acquittement (client). Un Consommateur de COS ("COS Consumer", Consommateur de changement d'état) est utilisé pour référencer l'appareil consommant les données de changement d'état ou cycliques et produisant un acquittement (serveur).

Production de données acquittées

a) Le chemin de connexion consommé des Producteurs de COS doit être mis à un objet Acknowledge Handler disponible. Le chemin doit être constitué de Class et Instance. Si un objet Acknowledge Handler n'est pas disponible, utiliser le service Create de la Classe Acknowledge Handler pour en obtenir un nouveau.

b) L'application E/S productrice des Producteurs de COS informe l'objet Acknowledge Handler de la production de nouvelles données (message d'événement Data Sent) ou des répétitions de tentative de production de données (message d'événement Data Resent).

c) La réception d'acquittement de Producteurs de COS est réalisée par l'objet Acknowledge Handler. L'objet Acknowledge Handler informe l'application E/S productrice lorsqu'un ou plusieurs acquittements n'ont pas été reçus dans les limites de temporisation d'acquittements (en utilisant les attributs Ack List et Ack Timeout).

d) Le message d'acquittement ne requiert aucune donnée. L'objet Acknowledge Handler de l'appareil producteur de COS doit considérer une réception de message valide comme étant un acquittement. Cependant, il est permis de configurer un appareil producteur de changement d'état ou cyclique pour consommer des données avec l'acquittement. Dans ce cas, les données sont transmises à l'objet d'application dans l'attribut "Data with Ack Path List", en se basant sur la connexion qui a reçu les données.

e) L'application productrice d'acquittement de Consommateur de COS doit être configurée pour envoyer un message de longueur zéro ou un message (de sortie) de réponse valide lorsqu'un message d'entrée valide est consommé.

f) Un temporisateur d'acquittement est déclenché chaque fois qu'une production survient. L'objet Acknowledge Handler reçoit notification de cet événement par un message d'événement Data Sent ou Data Resent provenant de l'application productrice.

g) L'expiration du temporisateur d'acquittements entraîne l'envoi d'un message Acknowledge Timeout vers l'objet d'application productrice. L'objet en question doit envoyer de nouveau le dernier message si la Retry Limit n'a pas été atteinte. Il peut également prendre une action spécifique à une application.

h) Le compte de répétitions de tentative est incrémenté chaque fois qu'un message Acknowledge Timeout est envoyé à l'application productrice. Lorsque la limite des répétitions de tentative a été atteinte, le message Retry Limit Reached est envoyé à l'objet d'application productrice.

i) Le compte de répétitions de tentative est vidé à chaque message Data Sent. Un message Data Resent ne vide pas le compteur de tentatives.

j) La valeur du temporisateur d'acquittements est configurable au sein de l'objet Acknowledge Handler.

k) Le nombre de répétitions de tentative est (facultativement) configurable au sein de l'objet Acknowledge Handler.

Utilisation de temporisateurs avec production de données acquittées

Les règles suivantes doivent être observées pour envoyer des données acquittées.

l) De nouvelles données ne sont pas envoyées pendant que l'Inhibit Timer est actif (en marche).

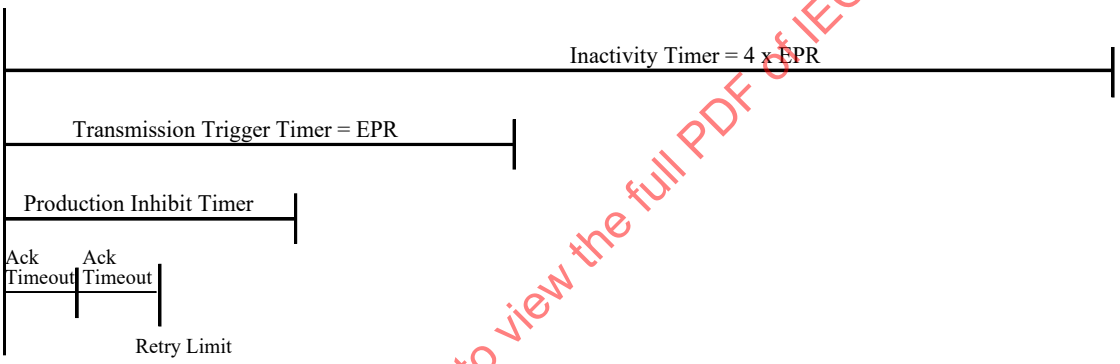
m) De nouvelles données sont envoyées lorsqu'aucun acquittement n'est en cours, en respectant la règle n°a (un acquittement est en cours après un envoi de nouvelles données ou une répétition de tentative d'anciennes données et jusqu'à un Ack Timeout ou Ack Received).

n) La répétition de tentative d'anciennes données se produit à Ack Timeout si de nouvelles données ne sont pas disponibles ou l'Inhibit Timer est actif.

o) L'envoi de nouvelles données (ou d'anciennes données sur temporisation de déclencheur d'émission) démarre l'Ack Timer, l'Inhibit Timer et le Transmission Trigger Timer. Le Retry Counter est également effacé.

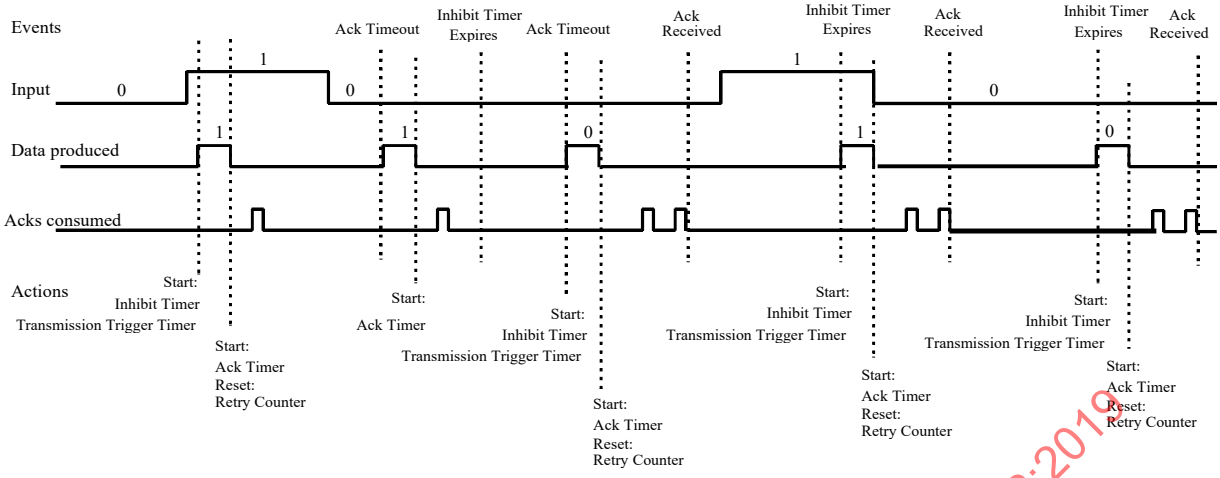
p) Une répétition de tentative d'anciennes données démarre l'Ack Timer.

La Figure 7 indique la relation typique des temporisateurs au sein de l'objet Acknowledge Handler. Un exemple de relations de temporisation types est montré à la Figure 8.



Anglais	Français
Inactivity Timer = 4 x EPR	Inactivity Timer = 4 x EPR
Transmission Trigger Timer = EPR	Transmission Trigger Timer = EPR
Production Inhibit Timer	Production Inhibit Timer
Ack Timeout	Temporisation d'acquittement
Retry Limit	Limite des répétitions de tentative

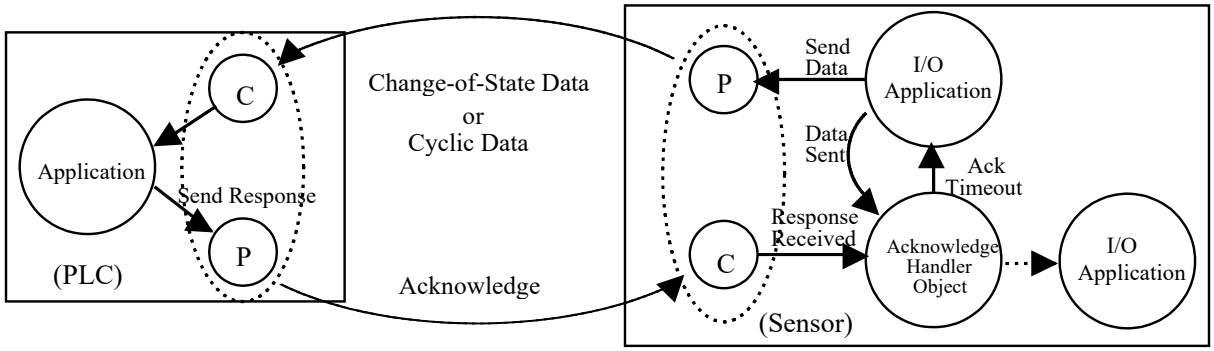
Figure 7 – Relations de temporisation types pour la production de données acquittées



Anglais	Français
Events	Evénements
Ack Timeout	Temporisation d'acquittement
Inhibit Timer Expires	Le Inhibit Timer expire.
Ack Received	Acquittement reçu
Input	Entrée
Data produced	Données produites
Acks consumed	Acquittements consommés
Actions	Actions
Start: Inhibit Timer Transmission Trigger Timer	Démarrer: Inhibit Timer Transmission Trigger Timer
Start: Ack Timer Reset: Retry Counter	Démarrer: Ack Timer Réinitialiser: Compteur des répétitions de tentative

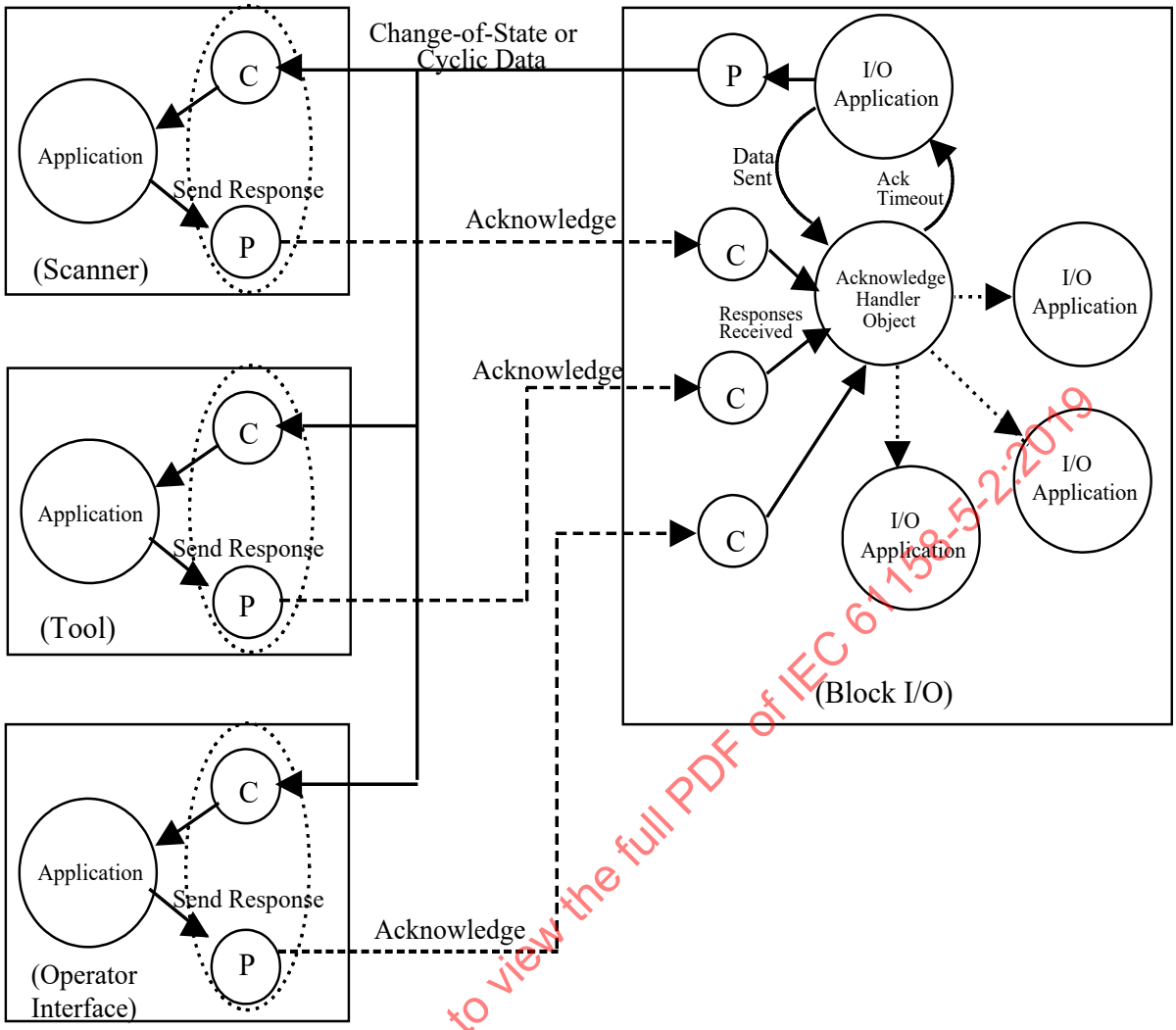
Figure 8 – Exemple de système COS doté de deux appareils d'acquittement

Les diagrammes suivants montrent le flux de messages dans une connexion Change of State pour des configurations à un seul consommateur (voir Figure 9) et à plusieurs consommateurs (voir Figure 10).



Anglais	Français
Application	Application
Send Response	Envoi de la réponse
Change-of-State Data or Cyclic Data	Données Change-of-State (changement d'état) ou Cyclic (cycliques)
Acknowledge	Acquittement
Send Data	Envoi des données
Data Sent	Données envoyées
I/O Application	Application E/S
Ack Timeout	Temporisation d'acquittement
Response Received	Réponse reçue
Acknowledge Handler Object	Objet Acknowledge Handler (gestionnaire d'acquittements)
(Sensor)	(Capteur)

Figure 9 – Flux de messages dans la connexion COS (un objet Connection, un consommateur)



Anglais	Français
Application	Application
Send Response	Envoi de la réponse
(Scanner)	(Dispositif de balayage)
Change-of-State Data or Cyclic Data	Données Change-of-State (changement d'état) ou Cyclic (cycliques)
Acknowledge	Acquittement
Data Sent	Données envoyées
I/O Application	Application E/S
Ack Timeout	Temporisation d'acquittement
Responses Received	Réponses reçues
Acknowledge Handler Object	Objet Acknowledge Handler (gestionnaire d'acquittements)
(Block I/O)	(Bloc E/S)
(Tool)	(Outil)
(Operator Interface)	(Interface opérateur)

Figure 10 – Flux de messages dans la connexion COS (plusieurs consommateurs)

6.2.1.2.6 Modèle formel pour Time Sync

6.2.1.2.6.1 Définition de classe

L'ASE (objet) Time Sync fournit une interface Type 2 IEC 61158 au protocole de synchronisation d'horloges de précision IEC 61588:2009 pour des systèmes de mesure et de commande mis en réseau, communément appelé le protocole de temps de précision (Precision Time Protocol (PTP)). Tout appareil prenant en charge la synchronisation du temps de type 2 doit fournir une seule instance (instance 1) de l'objet Time Sync.

NOTE Des détails complémentaires peuvent être consultés dans l'IEC 61588:2009.

Cet objet fournit des attributs et des services pour:

- récupérer le statut d'horloge et des propriétés telles que l'état synchronisé, le décalage courant par rapport à l'horloge mère, l'identité d'horloge grand-mère;
- accéder à des fonctions de gestion d'horloges PTP telles que la priorité d'horloge;
- accéder au réseau PTP d'appareils par le biais de messages de gestion PTP natifs.

Placé en face d'un attribut ou d'un service, le symbole (*) signifie que cet attribut ou ce service est soit obligatoire, soit facultatif, sur la base de certaines contraintes définies dans la description de l'attribut/du service.

FAL ASE:			**FAL Management ASE**
CLASS:			**Time_Sync_Object**
CLASS ID:			**67**
PARENT CLASS:			**Base_Object**
ATTRIBUTS D'ACCES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
ATTRIBUTS DE GESTION DE SYSTEME (ATTRIBUTS DE CLASSE):			
1	(m)	Attribute:	Revision = 3
2	(o)	Attribute:	Max Instance
3	(o)	Attribute:	Number of Instances
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
ATTRIBUTS DE GESTION D'OBJETS (ATTRIBUTS D'INSTANCE):			
1	(m)	Attribute:	PTPEnable
2	(m)	Attribute:	IsSynchronized
3	(m)	Attribute:	SystemTimeMicroseconds
4	(m)	Attribute:	SystemTimeNanoseconds
5	(m)	Attribute:	OffsetFromMaster
6	(m)	Attribute:	MaxOffsetFromMaster
7	(m)	Attribute:	MeanPathDelayToMaster
8	(m)	Attribute:	GrandMasterClockInfo
8.1	(m)	Attribute:	ClockIdentity
8.2	(m)	Attribute:	ClockClass
8.3	(m)	Attribute:	TimeAccuracy
8.4	(m)	Attribute:	OffsetScaledLogVariance

8.5	(m)	Attribute:	CurrentUtcOffset
8.6	(m)	Attribute:	TimePropertyFlags
8.7	(m)	Attribute:	TimeSource
8.8	(m)	Attribute:	Priority1
8.9	(m)	Attribute:	Priority2
9	(m)	Attribute:	ParentClockInfo
9.1	(m)	Attribute:	ClockIdentity
9.2	(m)	Attribute:	PortNumber
9.3	(m)	Attribute:	ObservedOffsetScaledLogVariance
9.4	(m)	Attribute:	ObservedPhaseChangeRate
10	(m)	Attribute:	LocalClockInfo
10.1	(m)	Attribute:	ClockIdentity
10.2	(m)	Attribute:	ClockClass
10.3	(m)	Attribute:	TimeAccuracy
10.4	(m)	Attribute:	OffsetScaledLogVariance
10.5	(m)	Attribute:	CurrentUtcOffset
10.6	(m)	Attribute:	TimePropertyFlags
10.7	(m)	Attribute:	TimeSource
11	(m)	Attribute:	NumberOfPorts
12	(m)	Attribute:	PortStateInfo
12.1	(m)	Attribute:	NumberOfPorts
12.2.1	(m)	Attribute:	PortNumber
12.2.2	(m)	Attribute:	PortState
13	(m)	Attribute:	PortEnableCfg
13.1	(m)	Attribute:	NumberOfPorts
13.2.1	(m)	Attribute:	PortNumber
13.2.2	(m)	Attribute:	PortEnable
14	(m)	Attribute:	PortLogAnnounceIntervalCfg
14.1	(m)	Attribute:	NumberOfPorts
14.2.1	(m)	Attribute:	PortNumber
14.2.2	(m)	Attribute:	PortlogAnnounceInterval
15	(m)	Attribute:	PortLogSyncIntervalCfg
15.1	(m)	Attribute:	NumberOfPorts
15.2.1	(m)	Attribute:	PortNumber
15.2.2	(m)	Attribute:	PortLogSyncInterval
16	(*)	Attribute:	Priority1
17	(*)	Attribute:	Priority2
18	(m)	Attribute:	DomainNumber
19	(m)	Attribute:	ClockType
20	(m)	Attribute:	ManufactureIdentity
21	(m)	Attribute:	ProductDescription
21.1	(m)	Attribute:	Size
21.2	(m)	Attribute:	Description
22	(m)	Attribute:	RevisionData
22.1	(m)	Attribute:	Size
22.2	(m)	Attribute:	Revision
23	(m)	Attribute:	UserDescription
23.1	(m)	Attribute:	Size
23.2	(m)	Attribute:	Description
24	(m)	Attribute:	PortProfileIdentityInfo
24.1	(m)	Attribute:	NumberOfPorts
24.2.1	(m)	Attribute:	PortNumber
24.2.2	(m)	Attribute:	PortProfileIdentity

25	(m)	Attribute:	PortPhysicalAddressInfo
25.1	(m)	Attribute:	NumberOfPorts
25.2.1	(m)	Attribute:	PortNumber
25.2.2	(m)	Attribute:	PhysicalProtocol
25.2.3	(m)	Attribute:	SizeOfAddress
24.2.4	(m)	Attribute:	PortPhysicalAddress
26	(m)	Attribute:	PortProtocolAddressInfo
26.1	(m)	Attribute:	NumberOfPorts
26.2.1	(m)	Attribute:	PortNumber
26.2.2	(m)	Attribute:	NetworkProtocol
26.2.3	(m)	Attribute:	SizeOfAddress
26.2.4	(m)	Attribute:	PortProtocolAddress
27	(m)	Attribute:	StepsRemoved
28	(m)	Attribute:	SystemTimeAndOffset
28.1	(m)	Attribute:	SystemTime
28.2	(m)	Attribute:	SystemOffset
29	(*)	Attribute:	AssociatedInterfaceObjects
29.1	(m)	Attribute:	NumberOfPorts
29.2.1	(m)	Attribute:	PortNumber
29.2.2	(m)	Attribute:	AssociatedObjectPathSize
29.2.3	(m)	Attribute:	AssociatedObject
SERVICES DE GESTION DE SYSTEME:			
1	(o)	Mgt Service:	Get_Attributes_All
14	(m)	Mgt Service:	Get_Attribute_Single
SERVICES DE GESTION D'OBJETS:			
3	(m)	Ops Service:	Get_Attribute_List
4	(m)	Ops Service:	Set_Attribute_List
14	(m)	Ops Service:	Get_Attribute_Single
16	(m)	Ops Service:	Set_Attribute_Single

6.2.1.2.6.2 Attributs de gestion de système (attributs de classe)

Aucune exigence spécifique pour cet objet.

6.2.1.2.6.3 Attributs de gestion d'objets

PTPEnable

Spécifie si, oui ou non, le Precision Time Protocol est activé pour cet appareil.

L'attribut PTPEnable ne s'applique pas à la ou aux fonctionnalités d'horloge transparente d'un appareil. Si les fonctions d'horloge transparente d'un appareil sont configurables, elles sont gérées par des moyens spécifiques à un fournisseur.

Les valeurs par défaut spécifiées dans le Tableau 15 ont été choisies pour simplifier la construction initiale du système dans la plupart des cas. Il y a des exigences conflictuelles pour les appareils mettant en œuvre une horloge frontière. Par exemple, si l'appareil est conçu pour jouer le rôle d'un maître CIP Sync (par exemple un dispositif de commande avec plusieurs ports PTP), il convient qu'il soit par défaut sur Disabled (Désactivé). Cependant, dans la mesure du possible, il convient que les horloges frontières soient sur Enabled (Activé) par défaut.

Tableau 15 – Valeurs par défaut de l'attribut PTPEnable

Type d'horloge PTP	Default Value
Horloge ordinaire esclave uniquement	Enabled (Activé)
Horloge frontière	Spécifique au produit, de préférence Enabled (Activé)
Horloge Ordinaire à capacité de maître	Disabled (Désactivé)
Horloge transparente	Non applicable

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

Toutefois, les appareils modulaires avec une horloge frontière peuvent considérer l'attribut PTPEnable comme volatile si un autre module fournit la valeur.

IsSynchronized

Spécifie si, oui ou non, l'horloge locale est synchronisée à l'horloge mère de référence. Une horloge est synchronisée, au minimum, s'il a un port dans l'état esclave et reçoit des mises à jour issues de la base de temps. Un appareil spécifique peut imposer des critères ou contraintes supplémentaires qui ne sont pas couverts par le présent document.

EXEMPLE Un appareil peut spécifier un seuil de synchronisation de 10 µs et indiquer seulement que l'appareil est synchronisé lorsque OffsetFromMaster (attribut 5) est inférieur à 10 µs pendant une durée de 5 intervalles de synchronisation (attribut 15).

Cet attribut d'instance volatil a une règle d'accès Get seulement.

SystemTimeMicroseconds

Spécifie l'heure système courante en microsecondes. Voir 6.2.1.2.6.5 (Définition de System Time).

Cet attribut d'instance volatil a une règle d'accès Get/Set. Set est obligatoire pour les appareils à capacité d'horloge maître et est facultatif dans le cas contraire.

SystemTimeNanoseconds

Idem que SystemTimeMicroseconds, excepté les unités de nanosecondes.

Cet attribut d'instance volatil a une règle d'accès Get/Set. Set est obligatoire pour les appareils à capacité d'horloge maître et est facultatif dans le cas contraire.

OffsetFromMaster

Spécifie la quantité d'écart entre l'horloge locale et son horloge mère en nanosecondes.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

MaxOffsetFromMaster

Spécifie la quantité maximale d'écart entre l'horloge locale et son horloge mère en nanosecondes depuis le dernier positionnement. La valeur peut être représentée comme valeur signée ou absolue. Elle est réglable, généralement sur 0.

Cet attribut d'instance volatil a une règle d'accès Get/Set.

MeanPathDelayToMaster

Spécifie le retard de chemin moyen entre l'horloge locale et son horloge mère en nanosecondes.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

GrandmasterClockInfo, ParentClockInfo, LocalClockInfo

GrandmasterClockInfo, ParentClockInfo et LocalClockInfo spécifient respectivement les informations de propriétés d'horloge pour l'horloge Grandmaster, Parent et Local PTP. Les données sont extraites des jeux de données PTP maintenues par le sous-système PTP.

L'attribut ClockIdentity fournit un identificateur unique pour l'horloge. La source de temps de l'horloge, sa classe, sa variance et autres attributs donnent des informations complémentaires concernant les propriétés de l'horloge.

Ces attributs d'instance volatils ont une règle d'accès Get seulement.

ClockIdentity

Spécifie identificateur unique pour l'horloge. Le format de l'identificateur dépend du protocole réseau. CP2/2 code l'adresse MAC en l'identificateur. CP2/3 et CP2/1 codent le Vendor ID et le Serial Number.

PortNumber

Spécifie le numéro de port de l'identité de port.

ClockClass

Spécifie la classe de la qualité de l'horloge. La classe d'horloge représente une mesure relative de la qualité de l'horloge utilisée par l'algorithme Best Master pour déterminer la grand-mère. La classe a une valeur entre 0 et 255, la meilleure horloge correspondant à 0.

TimeAccuracy

Spécifie l'exactitude absolue prévue de l'horloge par rapport à l'époque PTP. TimeAccuracy représente la mesure d'exactitude de la qualité de l'horloge utilisée par l'algorithme Best Master pour déterminer la grand-mère. L'exactitude est spécifiée comme étant une échelle graduée commençant à 25 ns (nanosecondes) et se terminant à une valeur supérieure à 10 s (secondes) ou inconnue. Une source de temps GPS aura approximativement une exactitude de 250 ns (nanosecondes). Une horloge Hand_Set aura typiquement une exactitude inférieure à 10 s (secondes). Plus l'exactitude est faible, meilleure est l'horloge.

OffsetScaledLogVariance

Spécifie une mesure des propriétés de stabilité intrinsèque de l'horloge. OffsetScaledLogVariance représente la mesure de variance de la qualité de l'horloge utilisée par l'algorithme Best Master pour déterminer la grand-mère. La valeur est représentée en unités logarithmiques normalisées de décalage. Plus la variance est faible, meilleure est l'horloge.

CurrentUtcOffset

Spécifie le décalage TUC courant, en secondes, par rapport au Temps atomique international (TAI) de l'horloge. Au 1er janvier 2006 à 00:00:00 TUC, le décalage était de 33 s (secondes).

TimePropertyFlags

Spécifie les fanions de propriétés de temps de l'horloge.

TimeSource

Spécifie la source de temps primaire de l'horloge.

Priority1 et Priority2

Spécifient la priorité relative de l'horloge grand-mère par rapport aux autres horloges dans le système. Voir respectivement Priority1 et Priority2, attributs principaux 16 et 17.

ObservedOffsetScaledLogVariance

Spécifie une mesure estimée de la variance de l'horloge parente telle qu'observée par l'horloge esclave.

ObservedPhaseChangeRate

Spécifie une mesure estimée de la dérive de l'horloge parente telle qu'observée par l'horloge esclave.

NumberOfPorts

Ce paramètre spécifie le nombre de ports PTP sur l'appareil. Les horloges PTP ordinaires ont un seul port. Les horloges PTP frontières et transparentes ont plusieurs ports. Une horloge hybride comprenant à la fois une horloge ordinaire et une horloge transparente de bout en bout a une valeur de un (1) pour cet attribut.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

Port stateInfo

Spécifie le statut courant de chaque port PTP sur l'appareil.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

PortEnableCfg

Spécifie la configuration d'activation de port relatif à chaque port PTP sur l'appareil.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

PortLogAnnounceIntervalCfg

Spécifie l'intervalle d'annonces PTP entre des messages “Announce” successifs émis par une horloge mère sur chaque port PTP de l'appareil. Les unités du membre PortLogAnnounceInterval sont des secondes en logarithme base 2. Voir 6.2.1.2.6.5 (Valeurs de réglages par défaut et plages de profil de PTP pour CPF 2) pour connaître les valeurs par défaut.

Cet attribut d'instance non volatil de structure a une règle d'accès Get/Set.

PortLogSyncIntervalCfg

Spécifie l'intervalle de synchronisation PTP entre des messages “Sync” successifs émis par une horloge mère sur chaque port PTP de l'appareil. Les unités du membre PortLogSyncInterval sont des secondes en logarithme base 2. Voir 6.2.1.2.6.5 (Valeurs de réglages par défaut et plages de profil de PTP pour CPF 2) pour connaître les valeurs par défaut.

Cet attribut d'instance non volatil de structure a une règle d'accès Get/Set.

Priority1

Spécifie la valeur de réglage Priority1 de l'algorithme Best Master PTP. Cet attribut spécifie la notation Best Master de cette horloge et annule et remplace la qualité d'horloge (classe, exactitude et variance). Cet attribut permet à l'utilisateur de neutraliser la sélection automatique de l'horloge "best master" (meilleure mère) avant que d'éventuelles mesures de qualité ne soient évaluées. La valeur est comprise entre 0 et 255. La priorité la plus élevée est 0. Voir 6.2.1.2.6.5 (Valeurs de réglages par défaut et plages de profil de PTP pour CPF 2) pour connaître les valeurs par défaut.

Cet attribut d'instance est obligatoire pour des appareils avec capacité de maître, il est facultatif pour tous les autres.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

Priority2

Spécifie la valeur de réglage Priority2 de l'algorithme Best Master PTP. Cet attribut spécifie la notation "Best Master" de cette horloge après que la qualité d'horloge (classe, exactitude et variance) a été évaluée et annule et remplace le tie-breaker . Cet attribut permet à l'utilisateur de neutraliser la sélection automatique de l'horloge "best master" (meilleure mère) après que des mesures de qualité ont été évaluées, c'est-à-dire de choisir la meilleure mère dans un ensemble d'horloges de même qualité. La valeur est comprise entre 0 et 255. La priorité la plus élevée est 0. Voir 6.2.1.2.6.5 (Valeurs de réglages par défaut et plages de profil de PTP pour CPF 2) pour connaître les valeurs par défaut.

Cet attribut d'instance est obligatoire pour des appareils avec capacité de maître, il est facultatif pour tous les autres.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

DomainNumber

Spécifie le domaine d'horloge PTP. Voir 6.2.1.2.6.5 (Valeurs de réglages par défaut et plages de profil de PTP pour CPF 2) pour connaître les valeurs par défaut.

Cet attribut d'instance non volatil a une règle d'accès Get/Set.

ClockType

Spécifie les fonctions PTP que le nœud prend en charge. Un nœud PTP peut mettre en œuvre plusieurs types d'horloges (par exemple, une horloge ordinaire peut être associée à une horloge transparente). Un indice de bit mis à 1 indique que le type d'horloge est pris en charge.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

ManufactureIdentity

Spécifie l'identité du fabricant de l'horloge. Les 3 premiers octets spécifient l'OUI (Organization Unique ID "identificateur propre à une organisation") de l'IEEE pour le fabricant. Le dernier octet est réservé.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

ProductDescription

Spécifie la description de produit de l'appareil qui contient l'horloge. Le format est:

- nom de fabricant de l'appareil suivi d'un point-virgule;

- numéro de modèle de l'appareil suivi d'un point-virgule;
- numéro de série.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

RevisionData

Spécifie la date de révision de l'appareil qui contient l'horloge. Le format est:

- révision du matériel de l'horloge suivie d'un point-virgule;
- révision du firmware de l'horloge suivie d'un point-virgule;
- révision du logiciel de l'horloge.

Les sous-trames qui ne s'appliquent pas peuvent être nulles ou vides (par exemple, "1.2;2.3;" ou ";;3.4").

Cet attribut d'instance volatil a une règle d'accès Get seulement.

UserDescription

Spécifie la description d'utilisateur de l'appareil qui contient l'horloge. Le format est:

- nom ou description, défini(e) par l'utilisateur, de l'appareil suivi(e) d'un point-virgule;
- emplacement physique, défini par l'utilisateur, de l'appareil.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

PortProfileIdentityInfo

Spécifie le profil PTP courant de chaque port de l'appareil. L'attribut retourne l'identité de profil relative au profil actif courant.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

PortPhysicalAddressInfo

Spécifie le protocole physique et l'adresse physique de chaque port de l'appareil.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

PortProtocolAddressInfo

Spécifie le protocole réseau et l'adresse protocole de chaque port de l'appareil (par exemple, adresse IP). Le NetworkProtocol spécifie le protocole pour le réseau.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

StepsRemoved

Spécifie le nombre de chemins de communication traversés entre l'horloge locale et l'horloge grand-mère

Cet attribut d'instance a une règle d'accès Get seulement.

SystemTimeAndOffset

Spécifie le System Time en microsecondes et l'Offset par rapport à la valeur d'horloge locale. L'appareil répondeur retournera les System Time et Offset courants. Voir le modèle d'horloge spécifique en 6.2.1.2.6.5.

Cet attribut d'instance volatil a une règle d'accès Get seulement.

AssociatedInterfaceObjects

Spécifie les objets associés à chaque port PTP de l'appareil.

Les numéros de port PTP doivent commencer par 1 et être séquentiels, conformément à l'IEC 61588. Lorsqu'il n'est pas possible que les numéros de port PTP et CPF 2 soient les mêmes, ou lorsque le port PTP est associé à un port physique, l'appareil doit identifier ces associations. L'attribut AssociatedInterfaceObjects spécifie pour chaque port PTP s'il est associé à un port CPF 2 ou à un port physique. Voir 6.2.1.2.6.5 c) (Attributions de ports PTP) pour des exemples où les numéros de ports PTP et CPF 2 peuvent ne pas être identiques.

Si le port PTP est associé à un port CPF 2, AssociatedObject doit spécifier l'instance d'objet qui représente le port CPF 2. En d'autres termes, AssociatedObject doit spécifier l'instance d'objet du port (20 F4 24 xx, où xx est l'instance d'objet du port).

Si le port PTP est associé à un port Ethernet physique (par exemple dans le cas PRP), AssociatedObject doit spécifier l'instance d'objet de liaison Ethernet (20 F6 24 xx, où xx est l'instance d'objet de liaison Ethernet).

Cet attribut d'instance est obligatoire si les numéros de port PTP ne correspondent pas aux numéros de port CPF 2, sinon il est facultatif.

Cet attribut d'instance non volatil a une règle d'accès Get seulement.

6.2.1.2.6.4 Services de gestion de système (niveau classe) et de gestion d'objets (niveau instance)

Aucune exigence spécifique pour cet objet.

6.2.1.2.6.5 Comportement spécifique pour l'objet Time Sync

a) Définition de System Time

La date/heure de début pour le System Time de la synchronisation du temps CPF2 correspond à 1970-01-01 (1er janvier 1970) à 00:00:00. Le System Time est représenté comme une valeur de 64 bits (ULINT) en microsecondes (attribut SystemTimeMicroseconds) ou nanosecondes (attribut SystemTimeNanoseconds). System Time est réajusté pour les secondes intercalaires, mais pas pour les fuseaux horaires locaux ou l'heure d'économie d'énergie. Il représente l'heure courante au méridien 0.

Dans le Precision Time Protocol, l'heure est distribuée comme le nombre de nanosecondes écoulées depuis le début de l'époque PTP, à savoir le 31 décembre 1969 à 23:59:51.999918 TUC. Les secondes intercalaires sont distribuées par le Precision Time Protocol sous la forme "décalage TUC courant". Au 1er janvier 2006 à 00:00:00 TUC, le nombre de secondes intercalaires était de 33.

L'heure et les secondes intercalaires PTP sont converties en microsecondes (attribut SystemTimeMicroseconds) ou nanosecondes (attribut SystemTimeNanoseconds) pour donner System Time comme étant:

SystemTime = PTPTime – CurrentUtcOffset

b) Identité d'horloge

Chaque horloge PTP doit affecter un identificateur unique de 8 octets pour l'horloge. Cette affectation d'identificateurs dépend du réseau CPF 2. Pour CP 2/2, les identificateurs sont basés sur l'adresse MAC. Pour les autres réseaux CPF 2, les identificateurs sont basés sur le Vendor Id et le Serial Number. Un bus local peut aussi mettre en œuvre le protocole PTP et affecter l'identificateur en utilisant le codage pour un réseau local ou fermé.

Les identificateurs sont formatés comme spécifié dans l'IEC 61158-6-2, Tableau 111.

c) Affectations de port PTP

Il est donné à chaque port réseau d'appareil un numéro de port PTP unique conforme à l'IEC 61588. Les numéros de port PTP doivent commencer par 1 et être séquentiels. Aux fins de cohérence avec les protocoles CPF 2, il est recommandé que les numéros de port PTP et CPF 2 soient identiques, si possible.

Les ports PTP et CPF 2 ne peuvent pas être les mêmes pour certaines raisons présentées ci-dessous.

- L'appareil peut ne pas numéroter ses ports CPF 2 séquentiellement à partir de 1. Par exemple, l'appareil peut ne pas prendre en charge le numéro 1 de port CPF 2 (port de fond de panier).
- Pour les appareils prenant en charge PRP, il doit y avoir deux ports PTP (un port par port Ethernet physique), mais un seul port CPF 2.

Voir 6.2.1.2.6.3 pour une description de l'attribut permettant de définir la relation de ports PTP.

d) Profil de PTP pour CPF 2

Comme défini par l'IEC 61588, le but d'un profil PTP profile est de permettre à des organisations de spécifier des sélections uniques de valeurs d'attributs et de caractéristiques facultatives du protocole qui, lors de l'utilisation du même protocole de transport, interfonctionneront et atteindront une performance qui satisfait aux exigences d'une application particulière.

Le profil PTP pour les réseaux CPF 2 est basé sur le profil PTP par défaut Delay Request-Response défini dans l'IEC 61588.

NOTE L'identification et les valeurs de réglages par défaut du profil PTP pour CPF 2 seront les mêmes que la spécification par défaut du PTP selon l'IEC 61588.

e) Identification de profil

Le Tableau 16 identifie le profil de PTP pour CPF 2.

Tableau 16 – Identification de profil

Identificateur de profil	Spécification
Description de profil de PTP	CIP Sync Profile
Version	1,0
Identificateur de profil	00-21-6C-00-01-00
Organisation de spécification	ODVA: Web: www.odva.org

f) Valeurs de réglages par défaut et plages de profil de PTP pour CPF 2

Le profil de CPF 2 doit prendre en charge les valeurs de réglages par défaut et plages pour PTP telles que définies dans le Tableau 17. Le tableau suivant définit les paramètres de plage exigés pour ce profil, mais un appareil peut prendre en charge une plage plus large (par exemple, un intervalle Log sync de -3 à +2) dans la mesure où ces valeurs sont conformes aux valeurs acceptées dans l'IEC 61588.

Tableau 17 – Valeurs de réglages par défaut et plages pour profil

Attribut PTP	ID d'attribut	Valeur par défaut [a]	Plage exigée	Plage acceptable	Units
Log announce interval	14	1	0 à 4	0 à 4	Log_2 secondes
Log sync interval	15	0	-1 à +1	-3 à +2	Log_2 secondes
Priority1	16	128	0 à 255	0 à 255	
Priority2	17	128	0 à 255	0 à 255	
Domain	18	0	0 à 127	0 à 127	
Announce receipt timeout interval		3	3	2 à 10	Intervalles d'annonce

[a] Les paramètres de profil par défaut peuvent différer pour les ports connectés à un fond de panier propriétaire ou à un bus.

g) Transports de profil

Le Tableau 18 identifie le transport IEC 61588 spécifié pour chaque réseau de type 2 et une référence correspondante dans la spécification IEC 61588.

Tableau 18 – Transports de profil

Réseau	Transport IEC 61588	Référence IEC 61588
CP 2/1	Transport de PTP via CP 2/1	Annexe H
CP 2/2	Transport de PTP via UDP/IPv4	Annexe D
CP 2/3	Transport de PTP via CP 2/3	Annexe G

h) Domaine de PTP

Un réseau CPF 2 doit être limité à un domaine PTP.

La spécification IEC 61588 prévoit la présence de plus d'un domaine d'horloge sur un réseau. Cependant, pour certaines mises en œuvre du PTP, basées sur le matériel IEC 61588:2004 (version 1), la présence d'autres domaines d'horloge provoquera une interférence. Par conséquent, pour éviter les problèmes d'interférence, un réseau CPF 2 sera limité à un seul domaine. Il est recommandé que les nouvelles mises en œuvre matérielles soient conçues pour tolérer la présence de plusieurs domaines.

i) Gestion de nœud PTP

Chaque appareil mettant en œuvre le profil PTP pour CPF 2 doit mettre en œuvre une instance de l'objet Time Sync. En outre, l'appareil doit mettre en œuvre le mécanisme de messages de gestion défini par l'IEC 61588.

j) Horloge transparente

Les appareils dotés de plusieurs ports entrée/sortie doivent mettre en œuvre de bout en bout une horloge transparente entre ces ports.

k) Mécanisme de retard de chemin

Chaque appareil mettant en œuvre le profil PTP pour CPF 2 doit mettre en œuvre le mécanisme de mesure de retard de chemin pour demande-réponse de retard.

l) Valeurs recommandées de réglage d'horloge PTP

Le Tableau 19 définit les valeurs recommandées de réglage par défaut pour les attributs d'horloge PTP. Ces valeurs de réglage donnent des recommandations générales pour que divers types d'appareils affectent une priorité relative utilisée par l'algorithme d'horloge "meilleure mère" ("Best Master Clock Algorithm" (BMCA)) du protocole PTP. Typiquement, les appareils de commande, les commutateurs, les sources deviendront des maîtres sur E/S parce qu'ils ont de meilleures valeurs de réglage.

Les appareils à capacité de grand-mère peuvent réajuster mes valeurs de réglage des classes d'horloges, de l'exactitude d'horloge et des sources de temps. Par exemple, une horloge GPS, changera de façon dynamique sa classe et son exactitude d'horloge une fois qu'elle a verrouillé sur des satellites. Les valeurs de réglages de Priority peuvent être modifiées sur un appareil particulier pour neutraliser le comportement de sélection de Best Master Clock.

Tableau 19 – Valeurs de réglage par défaut d'horloge PTP

Attribut d'horloge PTP	Dispositif de commande	Appareil E/S	Commutateur ou routeur	Source primaire
Clock Type(s) (Type(s) d'horloge)	Ordinaire (Maître / Esclave)	Ordinaire (Esclave seulement)	Frontière et/ou Transparente	Ordinaire (Maître seulement)
Clock Class (Classe d'horloge)	248	255	248	248
Clock Accuracy (Exactitude d'horloge)	Inconnu	Inconnu	Inconnu	Inconnu
Clock Variance (Variance d'horloge)	Spécifique à l'appareil	Spécifique à l'appareil	Spécifique à l'appareil	Spécifique à l'appareil
Time Source (Source de temps)	Oscillateur	Oscillateur	Oscillateur	Oscillateur
Priority1	128	128 (non réglable)	128	128
Priority2	128	128 (non réglable)	128	128

m) TTL – Time to Live (durée de vie, CP 2/2 seulement)

Par défaut, la valeur de TTL dans le paquet UDP/IPV4 doit être mise à 1 sur les réseaux CP 2/2. Ceci est requis afin d'empêcher que des paquets PTP ne traversent des routeurs susceptibles d'augmenter considérablement la gigue des paquets. Lorsque cela est exigé, l'heure PTP peut être faite pour traverser des sous-réseaux par l'utilisation de routeurs avec des horloges frontières de l'IEC 61588.

n) Gestion de la qualité des horloges Hand_Set

Certaines horloges grands-mères de CPF 2 permettent de fixer l'heure directement par l'utilisateur ou par l'intermédiaire d'un mécanisme administré par l'utilisateur. Par exemple, un appareil peut fournir une option de configuration pour régler le System Time depuis un ordinateur personnel (PC). Ce comportement est caractérisé comme étant une horloge Hand_Set. Il convient que le processus de réglage manuel d'une horloge améliore la qualité de l'horloge, car la nouvelle heure sera présumée être une référence plus exacte. Une horloge peut, par exemple, se mettre sous tension avec un temps inconnu. Une fois réglée, l'heure est connue, et ce, avec une tolérance spécifiée du temps TUC.

Le Tableau 20 définit les valeurs de réglage PTP pour une horloge qui prend en charge le comportement Hand_Set. Ces valeurs de réglage sont définies de sorte que toutes les horloges mettant en œuvre le comportement Hand_Set interfonctionneront. Les combinaisons classe d'horloge et exactitude d'horloge sont identifiées avec la source de temps. La source de temps est un identificateur d'informations seulement et ne constitue pas un facteur dans le choix de la meilleure mère.

Tableau 20 – Gestion de la qualité des horloges Hand_Set

Etat d'horloge	Classe d'horloge	Exactitude d'horloge	Source de temps
Immédiatement après réglage manuel	Degradated Reference (B) (187)	A 10 s (secondes) près (0x30)	Hand_Set (0x60)
Mise sous tension au déballage	Default (248)	Inconnue (0xFE)	Oscillateur (0xA0)
NOTE Une référence dégradée B (187) est choisie pour la classe d'horloge au lieu d'une classe de référence primaire (6) ou de référence Holdover (7) parce que la référence dégradée B permet que l'esclave soit un esclave lorsqu'une meilleure mère est choisie. Plus précisément, cela signifie que l'horloge Hand_Set synchronisera son horloge à l'horloge grand-mère courante plutôt que de passer à un état de mère PASSIVE . De plus, le vieillissement de la qualité de l'horloge au fil du temps n'est pas pris en considération – par exemple, la dégradation à une horloge de qualité inférieure à mesure que l'exactitude dérive au fil du temps, mais un tel comportement est acceptable.			

o) Prise en charge de la reconfiguration de Device Level Ring et de chemin

Les appareils qui prennent en charge les topologies en anneau et qui nécessitent une précision de synchronisation élevée doivent prendre en charge le recalcul automatique du chemin à chaque fois que la topologie en anneau est modifiée. Ces appareils surveilleront directement le réseau pour détecter tout changement dans la topologie du réseau ou en seront indirectement informés par d'autres nœuds chargés de surveiller le réseau. La passerelle DLR et les nœuds d'extrémité DLR doivent mettre en œuvre le message CIP Sync Path Reconfiguration Signalling (voir 6.2.1.2.6.5, p)).

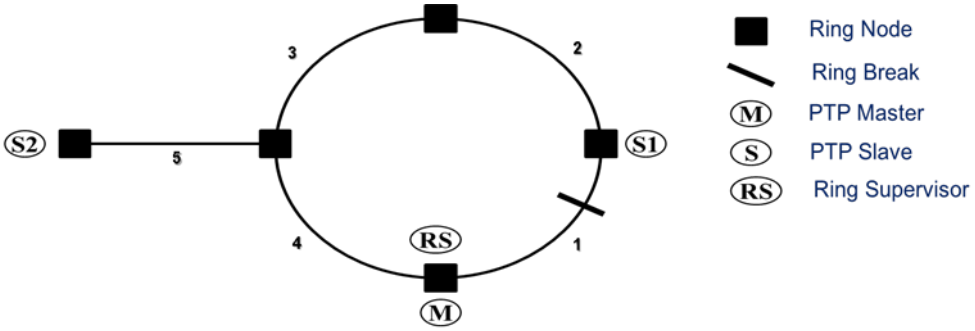
NOTE Voir l'Article 10 de l'IEC 61158-4-2 pour plus d'informations sur la prise en charge de Device Level Ring (DLR) et de CIP Sync.

Dans DLR, tout changement de la topologie nécessite que les appareils esclaves CIP Sync recalculent la mesure de retard de chemin vers le maître lorsque le chemin est modifié. Par exemple, une rupture dans la boucle entraînera la modification du chemin et de la mesure de chemin correspondante entre le maître et l'esclave. Il est recommandé de terminer ce calcul le plus rapidement possible, à savoir dans les quelques cycles de synchronisation suivant la détection d'une reconfiguration du réseau.

Un appareil PTP esclave détecte que le chemin a été modifié par l'un des deux mécanismes. Un appareil esclave dans la boucle détectera que la boucle a été interrompue, comme l'indique le statut de la boucle dans le protocole de la boucle. Un appareil esclave en dehors de la boucle détectera que le chemin a été modifié lorsqu'il reçoit un message CIP Sync Path Reconfiguration Signalling de la part du nœud de pont.

Il convient que les appareils de pont qui relient une boucle et un réseau externe génèrent le message Signalling sur le réseau en pont non bouclé lorsqu'une rupture de boucle est détectée et que le maître se trouve dans la boucle.

Ces deux scénarios sont représentés à la Figure 11. Le chemin entre le maître M et l'esclave S1 avant la rupture de la boucle est le chemin étiqueté 1. Après la rupture de la boucle, ce chemin correspond à 2 + 3 + 4. Le nœud S1 détectera la rupture de la boucle et recalculera le chemin. Le chemin entre le maître M et l'esclave S2 avant la rupture de la boucle correspond à 1 + 2 + 3 + 5. Après la rupture de la boucle, ce chemin correspond à 4 + 5. L'appareil de passerelle DLR entre les chemins 4 et 5 détectera la rupture de la boucle et signalera au nœud S2 de recalculer le chemin.



Anglais	Français
Ring Node	Nœud de la boucle
Ring Break	Rupture de la boucle
PTP Master	Maître PTP
PTP Slave	Esclave PTP
Ring Supervisor	Superviseur de la boucle

Figure 11 – Reconfiguration de chemin dans une topologie en boucle

p) Message Path Reconfiguration Signalling

Le message Path Reconfiguration Signalling est émis par un appareil lorsqu'il détecte une reconfiguration du réseau. Ce message est un message Signalling spécifique au profil CIP Sync. Les horloges mères en boucle et tolérantes aux pannes, ainsi que d'autres appareils réseau, émettent ce message lorsqu'ils détectent une reconfiguration du réseau. Lorsqu'un nœud reçoit ce message, il est recommandé qu'il recalcule son retard de chemin moyen à l'aide du mécanisme de demande-réponse de retard. Les détails du message Signalling sont indiqués dans le Tableau 21 et l'IEC 61588.

Tableau 21 – Message Path Reconfiguration Signalling

TLV Signalling	Description du champ	Valeur du champ
Type TLV	Extension de l'organisation	03
ID organisation	ID CIP Sync	00 21 6c
Sous-type TLV	Reconfiguration du chemin	01

q) Modèle d'horloge

La synchronisation du temps de CPF2 définit un modèle d'horloge décalé pour répondre aux exigences pour les applications de commande industrielles.

NOTE Ce modèle est nécessaire, car, bien que le protocole PTP de l'IEC 61588:2009 PTP définisse un mécanisme pour distribuer et synchroniser le temps, il ne définit pas un mécanisme pour compenser les changements par échelons du temps qui pourraient se produire à la source d'horloge grand-mère.

Les changements par échelons du temps peuvent survenir en raison d'une ou plusieurs des conditions suivantes:

- L'utilisateur réajuste l'horloge mère dont le type est Hand_Set.
- Un maître avec une horloge plus exacte devient disponible (nouvelle grand-mère). Cela peut se produire pendant le démarrage du système ou après que le système a fonctionné pendant un certain temps.

- La base de temps est temporairement déconnectée de l'horloge esclave puis est reconnectée. Dans cette situation, étant donné une discordance de temps entre le maître et l'esclave, il se produira un changement par échelon.

Il convient que les applications nécessitant une horloge stable en présence de changements par échelons du temps mettre en œuvre leur horloge PTP comme une horloge locale, synchronisée à la fréquence de l'horloge mère PTP, mais pas à la valeur du temps absolu de l'horloge mère PTP.

Dans ce modèle, le protocole PTP est utilisé pour discipliner l'horloge locale afin qu'elle fournisse des tops à la même fréquence que la mère. L'appareil maintient alors un décalage entre l'heure d'horloge locale et l'heure système. Une faible variation delta du temps amènera l'appareil à apporter un petit réajustement au décalage et à continuer de "régler finement" son horloge. Une forte variation du temps amènera l'appareil à mettre à jour son décalage, mais pas à "régler finement" son horloge.

Des événements cycliques peuvent alors être programmés en se basant sur l'horloge locale et ne seront pas altérés par de fortes variations par échelons du temps — à condition que l'horloge locale reste synchronisée à la fréquence de l'horloge mère PTP.

La Figure 12 montre la relation entre Local Clock, System Time Offset et System Time pour l'esclave PTP Time Slave. Le “system to local clock offset” (décalage d'horloge système par rapport à l'horloge locale) est une variable de mémoire maintenue par la mise en œuvre de l'horloge PTP pour effectuer des réajustements de décalage. Une valeur en secondes intercalaires est ajoutée au décalage pour donner le System Time en termes de l'époque correcte.

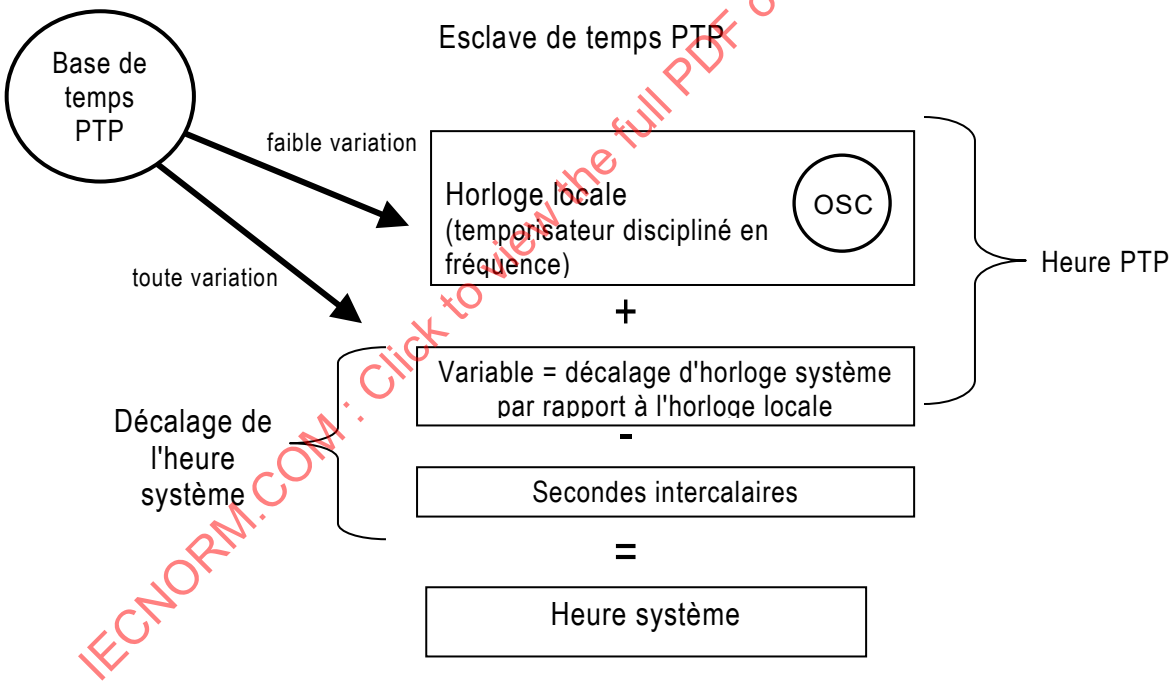


Figure 12 – Modèle d'horloge de décalage de la synchronisation du temps CPF2

La Figure 13 montre un système d’appareils qui mettent chacun en œuvre le Modèle d'horloge de décalage (Offset Clock Model) pour synchronisation du temps CPF2. Chaque appareil est partie intégrante d'un réseau d’appareils qui partagent la même notion de System Time. Chaque appareil a également une valeur de Local Clock qui est basée sur un temporisateur discipliné en fréquence et qui est reliée au System Time par une valeur de décalage (System Time Offset). Un tel système permet à chaque appareil de maintenir une heure locale indépendante vis-à-vis de tous les autres appareils, mais de partager une même notion d'heure système. A ce titre, le System Time peut changer sans nécessiter des changements de l'horloge locale.

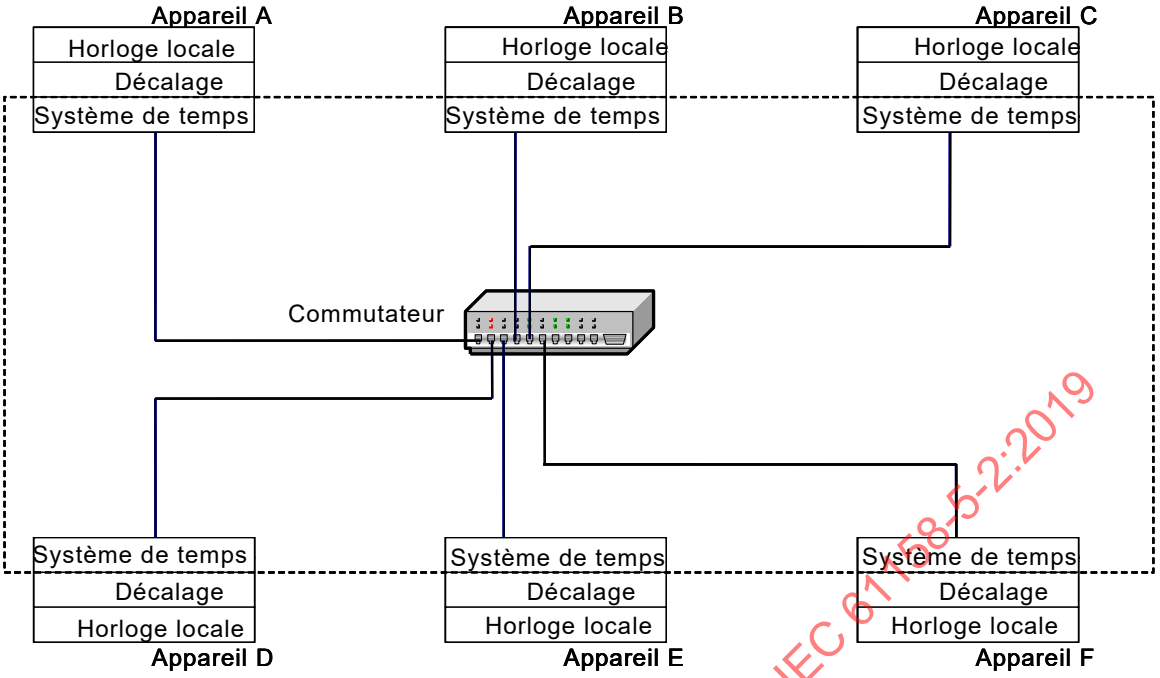


Figure 13 – Système de synchronisation du temps CPF2 avec modèle d'horloge de décalage

r) Compensation d'heure système

Une application crée un marqueur temporel (timestamp) en lisant l'horloge qui a été synchronisée par le PTP et en appliquant tous les réajustements requis à la valeur, par exemple en la convertissant en System Time. Des réajustements complémentaires peuvent être requis s'il s'est produit un échelon de System Time.

La synchronisation de temps CPF2 définit un mécanisme pour maintenir un jeu cohérent de marqueurs temporels en présence de changements par échelons dans le System Time. Un changement par échelons dans le System Time produit effectivement un décalage de la base de temps du système. Tout changement par échelon de l'heure d'horloge grand-mère doit se propager à travers le système. Sachant que tous les nœuds dans le système ne verront pas le changement par échelons au même instant, un marqueur temporel pris sur un nœud peut être incompatible avec un marqueur temporel sur un autre nœud. Ou bien un marqueur temporel pris à un instant donné peut être incompatible avec un marqueur temporel pris à un instant plus tardif sur le même nœud. Un algorithme de compensation est nécessaire pour rendre les marqueurs temporels compatibles les uns avec les autres avant de les utiliser dans des calculs.

Deux algorithmes possibles sont décrits dans les sections ci-après.

La décision de compenser les marqueurs temporels est prise en fonction des exigences de l'application.

s) Algorithmes de compensation par échelons

Une valeur de décalage est maintenue dans chaque marqueur temporel. Cette valeur de décalage est le System Time Offset (décalage de l'heure système) au moment où le marqueur temporel est saisi. Le décalage peut être utilisé pour fournir une indication relative au moment où il se produit un échelon dans le System Time. Si le marqueur temporel est émis à travers le réseau d'un nœud vers un second nœud, le décalage est envoyé en accompagnant le marqueur temporel. Si deux marqueurs temporels sont saisis au sein du même nœud, les décalages seront stockés avec les marqueurs temporels.

Les algorithmes transforment un marqueur temporel d'une base de temps en une seconde base de temps s'il se produit un changement de base de temps entre les instants auxquels les deux marqueurs temporels sont saisis. Lorsque les deux marqueurs temporels sont comparés, ils référenceront la même base de temps. Ces algorithmes peuvent également être appliqués à un jeu de marqueurs temporels.

t) Algorithme de compensation de marqueurs temporels au sein d'un seul nœud

Cet algorithme est défini pour permettre à un nœud de réajuster la valeur d'un ou plusieurs marqueurs temporels qui ont été saisis sur le nœud local.

L'algorithme est énoncé comme suit:

$$Timestamp_C = Timestamp_0 + Offset_1 – Offset_0$$

où:

$Timestamp_C$ est le marqueur temporel compensé;

$Timestamp_0$ est le marqueur temporel devant être compensé;

$Offset_1$ est le décalage associé au marqueur temporel à la base de temps cible;

$Offset_0$ est le décalage pour le marqueur temporel devant être compensé.

u) Algorithme de compensation pour marqueurs temporels entre nœuds

Cet algorithme est défini pour permettre à un nœud de réajuster la valeur d'un marqueur temporel reçu en provenance d'un nœud-source distant afin qu'il puisse être comparé à un marqueur temporel sur son propre nœud, le nœud-destination.

Le nœud-destination doit être capable de détecter un changement par échelon du System Time sur le nœud distant ou sur son propre nœud et effectuer le réajustement approprié.

Deux conditions sont possibles:

- l'appareil source a vu un changement par échelons du temps, mais l'appareil de destination ne l'a pas vu; ou
- l'appareil de destination a vu un changement par échelons du temps, mais l'appareil source ne l'a pas vu.

Le changement par échelons est détecté par un changement du SystemTimeOffset par la source ou par la destination. Le décalage source est envoyé à l'appareil de destination avec le marqueur temporel. L'appareil de destination compare le décalage reçu au décalage précédemment reçu afin de déterminer si un changement par échelon s'est produit et réajuste en conséquence la valeur du marqueur temporel reçu.

L'algorithme est énoncé comme suit:

$$TimestampCompensated = Timestamp_{Received} + ((Dest_{Offset} – Dest_{LastOffset}) – (Source_{Offset} – Source_{LastOffset}))$$

où:

$TimeStamp_{Received}$ est le marqueur temporel reçu;

$Dest_{Offset}$ est la valeur courante du décalage d'heure de l'horloge locale au niveau de la destination;

$Dest_{LastOffset}$ est la précédente valeur du décalage d'heure de l'horloge locale au niveau de la destination;

$Source_{Offset}$ est la valeur reçue du décalage d'heure de l'horloge locale par rapport à la source;

$Source_{LastOffset}$ est la précédente valeur du décalage d'heure de l'horloge locale par rapport à la source.

Les derniers décalages sont mis à jour lorsque:

$$|(Dest_{Offset} – Dest_{LastOffset}) – (Source_{Offset} – Source_{LastOffset})| <= SyncBoundsLimit$$

où:

SyncBoundsLimit est un nombre relativement faible qui définit ces bornes de synchronisation pour l'application.

v) Séquence de démarrage de groupe

La synchronisation de temps CPF2 définit une séquence de démarrage de groupe et un service de synchronisation de groupe. La séquence de démarrage de groupe permet à un groupe d'appareils de garantir que leurs horloges ont toutes été synchronisées à l'horloge mère de référence PTP avant de démarrer l'application qui dépend du System Time. L'application définit les exigences spécifiques nécessaires pour que le groupe soit considéré comme étant synchronisé.

Chaque application (objet) qui souhaite se synchroniser à un groupe doit mettre en œuvre le service Group_Sync.

Une application peut envoyer des paramètres complémentaires comme partie intégrante du service Group_Sync. Par exemple, elle peut échanger l'attribut SystemTimeOffset pour faciliter la compensation de marqueurs temporels, ou une période et une phase pour déclencher un événement périodique synchronisé, ou un ou plusieurs paramètres limites pour spécifier une exigence de synchronisation cible.

Le service retourne "true" si l'appareil est synchronisé à l'horloge mère PTP Time, sinon il retourne "false."

Un exemple de séquence de démarrage pour un groupe d'applications qui ont besoin de se synchroniser est montré à la Figure 14.

Une fois que le dispositif de commande lui-même est synchronisé à 'horloge mère, il déclenche une série de messages Group_Sync à destination de chacun des appareils faisant également partie du même groupe. Chaque appareil retourne une réponse indiquant si, oui ou non, il est également synchronisé à l'horloge mère de référence PTP. Si le dispositif de commande et tous les appareils sont synchronisés, le groupe est synchronisé. Autrement, l'application-mère répète la séquence de synchronisation ou prend une action de cas de défauts.

Le statut "synchronisé" de chaque nœud est déterminé par les exigences de l'application et peut être conditionné sur des paramètres de synchronisation complémentaires.

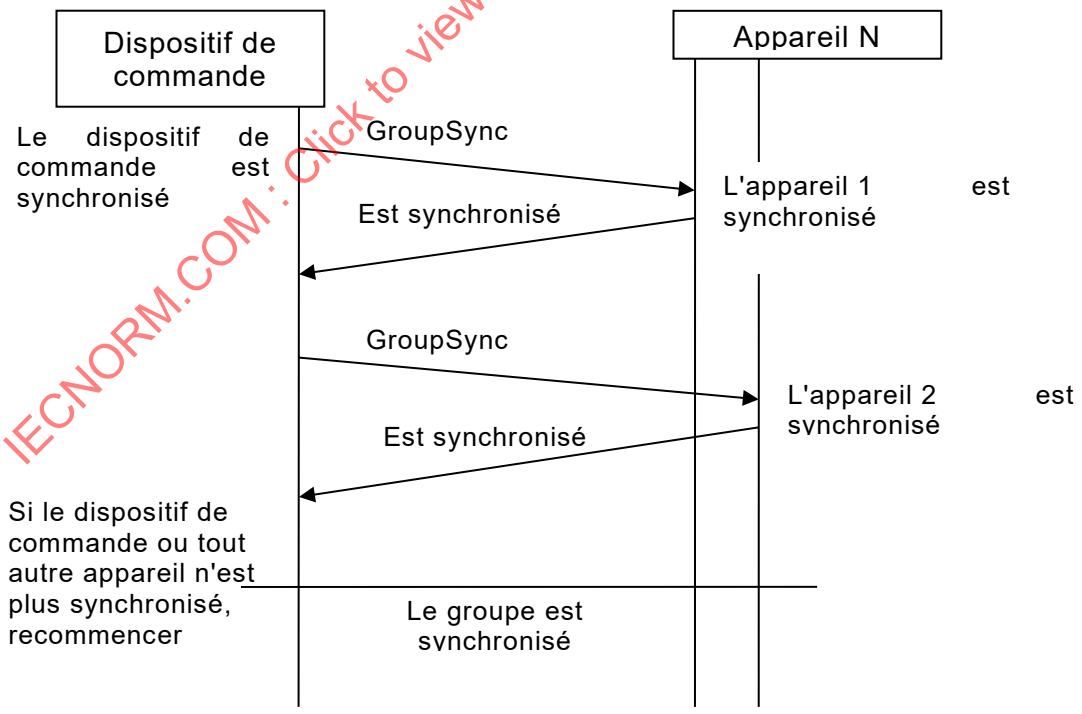


Figure 14 – Séquence de démarrage de groupe pour la synchronisation de temps

6.2.1.2.7 Modèle formel pour Parameter

6.2.1.2.7.1 Définition de classe

L'utilisation de l'objet Parameter fournit une interface publique connue à des données de configuration d'un appareil. De plus, cet objet donne également toutes les informations nécessaires pour définir et décrire chacun des paramètres de configuration individuels de l'appareil.

Cet objet permet à un appareil d'identifier complètement un paramètre configurable en fournissant une description complète du paramètre, y compris les valeurs minimale et maximale et une chaîne textuelle lisible par l'homme décrivant le paramètre.

Il doit y avoir une instance de cette classe d'objets pour chacun des paramètres configurables de l'appareil. Les instances doivent commencer à l'instance 1 sans le moindre trou dans les instances.

Chaque instance est liée à l'un des paramètres configurables, qui est typiquement (mais ce n'est pas obligatoire) un attribut de l'un des autres objets de l'appareil. Le fait de modifier l'attribut Parameter Value d'un objet Parameter se traduit par un changement correspondant de la valeur d'attribut indiquée par l'attribut Link Path (la valeur d'attribut sur laquelle pointe l'objet Parameter).

Un Parameter Object Stub est une version raccourcie d'un objet Parameter. Le Stub stocke seulement la valeur des données de configuration, fournissant seulement un point d'accès normalisé pour le paramètre.

Placé en face d'un attribut ou d'un service, le symbole (*) signifie que cet attribut ou ce service est soit obligatoire, soit facultatif, sur la base de certaines contraintes définies dans la description de l'attribut/du service.

FAL ASE:			**FAL Management ASE**
CLASS:			**Parameter_Object**
CLASS ID:			**15**
PARENT CLASS:			**Base_Object**
ATTRIBUTS D'ACCES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
ATTRIBUTS DE GESTION DE SYSTEME (ATTRIBUTS DE CLASSE):			
1	(o)	Attribute:	Revision = 1
2	(o)	Attribute:	Max Instance
8	(m)	Attribute:	Parameter Class Descriptor
9	(m)	Attribute:	Configuration Assembly Instance
10	(o)	Attribute:	Native Language
ATTRIBUTS DE GESTION D'OBJETS (ATTRIBUTS D'INSTANCE):			
1	(m)	Attribute:	Parameter Value
2	(m)	Attribute:	Link path size
3	(m)	Attribute:	Link path
4	(m)	Attribute:	Descriptor
5	(m)	Attribute:	Data Type
6	(m)	Attribute:	Data Size
7	(*)	Attribute:	Parameter Name String
8	(*)	Attribute:	Units String
9	(*)	Attribute:	Help String
10	(*)	Attribute:	Minimum Value
11	(*)	Attribute:	Maximum Value

12	(*)	Attribute:	Default Value
13	(*)	Attribute:	Scaling Multiplier
14	(*)	Attribute:	Scaling Divisor
15	(*)	Attribute:	Scaling Base
16	(*)	Attribute:	Scaling Offset
17	(*)	Attribute:	Multiplier Link
18	(*)	Attribute:	Divisor Link
19	(*)	Attribute:	Base Link
20	(*)	Attribute:	Offset Link
21	(*)	Attribute:	Decimal Precision
22	(o)	Attribute:	International Parameter Name
23	(o)	Attribute:	International Engineering Units
24	(o)	Attribute:	International Help String

SERVICES DE GESTION DE SYSTEME:

1	(o)	Mgt Service:	Get_Attributes_All
5	(o)	Mgt Service:	Reset
13	(o)	Mgt Service:	Apply_Attributes
14	(m)	Mgt Service:	Get_Attribute_Single
16	(o)	Mgt Service:	Set_Attribute_Single
21	(*)	Mgt Service:	Restore
22	(*)	Mgt Service:	Save

SERVICES DE GESTION D'OBJETS:

1	(*)	Ops Service:	Get_Attributes_All
13	(o)	Ops Service:	Apply_Attributes
14	(m)	Ops Service:	Get_Attribute_Single
16	(m)	Ops Service:	Set_Attribute_Single
21	(*)	Ops Service:	Restore
22	(*)	Ops Service:	Save
24	(*)	Ops Service:	Get_Member

SERVICES SPECIFIQUES A L'OBJET:

75	(o)	Ops Service:	Get_Enum_String

6.2.1.2.7.2 Attributs de gestion de système (attributs de classe)

Revision

Le niveau de révision de cet objet.

Cet attribut d'instance a une règle d'accès Get.

Max Instance

Numéro d'instance maximal d'un objet actuellement créé à ce niveau de classe de l'appareil.

Cet attribut d'instance a une règle d'accès Get.

Parameter Class Descriptor

Bits qui décrivent les paramètres.

Cet attribut d'instance a une règle d'accès Get.

Configuration Assembly Instance

Numéro d'instance de l'assemblage de configuration.

Cet attribut d'instance a une règle d'accès Get.

Native Language

Identificateur de langue pour tous les accès de matrice de caractères.

Si l'attribut Active Language de l'objet Identity (voir 6.2.1.2.2.3) est pris en charge, cet attribut ne doit pas être pris en charge.

Cet attribut d'instance a une règle d'accès Get/Set.

6.2.1.2.7.3 Attributs de gestion d'objets

Plusieurs attributs sont obligatoires, facultatifs ou non autorisés, sur la base de l'une des trois contraintes suivantes. La contrainte applicable est indiquée dans la description de service de l'attribut. Ils dépendent tous de la définition ou non du bit 1 dans le Parameter Class Descriptor (attribut de classe 8) (qui prend en charge les attributs complets).

Contrainte (1): Si le bit est défini, cet attribut est obligatoire, sinon il n'est pas autorisé.

Contrainte (2): Si le bit est défini, cet attribut est obligatoire, facultatif ou non autorisé, selon le type de données du paramètre, tel que spécifié dans l'IEC 61158-6-2, 4.1.8.7.1.2, sinon il n'est pas autorisé.

Contrainte (3): Si le bit 1 est défini, cet attribut est facultatif, sinon il n'est pas autorisé.

Parameter Value

Valeur effective du paramètre. Elle peut être accessible en lecture ou en écriture. La mise à l'échelle ne doit pas être effectuée pour cet attribut par l'objet Parameter. La mise à l'échelle, si elle est activée, doit être gérée par l'outil de configuration.

Cet attribut d'instance volatil a une règle d'accès Get/Set. Set est obligatoire si le bit de Read Only Parameter (4) dans l'attribut Descriptor (4) est FALSE (voir l'IEC 61158-6-2, 4.1.8.7.1.2). Cet attribut est utilisé dans l'objet Parameter Stub (raccourci) et Full (complet).

Link path size

Taille du chemin de liaison.

Cet attribut d'instance non volatil a une règle d'accès Get/Set. Cet attribut est utilisé dans l'objet Parameter Stub (raccourci) et Full (complet).

Link path

Chemin CIP menant à l'objet à partir duquel cette valeur de paramètre est récupérée.

Cet attribut d'instance non volatil a une règle d'accès Get/Set. Cet attribut est utilisé dans l'objet Parameter Stub (raccourci) et Full (complet).

Descriptor

Description du paramètre

Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Stub (raccourci) et Full (complet).

Data Type

Code de type de données.

Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Stub (raccourci) et Full (complet).

Data Size

Nombre d'octets dans l'attribut "Parameter Value".

Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Stub (raccourci) et Full (complet).

Parameter Name String

Chaîne lisible par l'homme qui représente le nom du paramètre.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Units String

Engineering Unit String (chaîne d'unités d'étude).

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Help String

Chaîne d'aide.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Minimum Value

La valeur minimale à laquelle le paramètre peut être mis.

La contrainte (2) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Maximum Value

La valeur maximale à laquelle le paramètre peut être mis.

La contrainte (2) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Default Value

La valeur effective à laquelle il convient de régler le paramètre lorsque l'utilisateur souhaite la valeur par défaut pour le paramètre.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Scaling Multiplier

Multiplicateur pour le Scaling Factor (Facteur de mise à l'échelle).

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Scaling Divisor

Diviseur pour la Scaling Formula (Formule de mise à l'échelle).

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Scaling Base

Base pour la Scaling Formula (Formule de mise à l'échelle).

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Scaling Offset

Décalage pour la Scaling Formula (Formule de mise à l'échelle).

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Multiplier Link

Instance de Parameter de la source Multiplier.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Divisor Link

Instance de Parameter de la source Divisor.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Base Link

Instance de Parameter de la source Base.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Offset Link

Instance de Parameter de la source Offset.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

Decimal Precision

Spécifie le nombre de décimales à utiliser pour afficher la valeur d'étude mise à l'échelle. Utilisé également pour déterminer la valeur d'incrément effective faisant que le fait d'incrémenter une valeur entraîne un changement de la valeur d'étude mise à l'échelle avec cette précision.

La contrainte (1) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

International Parameter Name

Chaîne lisible par l'homme qui représente le nom du paramètre.

La contrainte (3) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

International Engineering Units

Unités d'étude associées au paramètre.

La contrainte (3) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

International Help String

Chaîne d'aide.

La contrainte (3) s'applique. Cet attribut d'instance non volatil a une règle d'accès Get. Cet attribut est utilisé dans l'objet Parameter Full (complet).

6.2.1.2.7.4 Services de gestion de système (niveau classe) et de gestion d'objets (niveau instance)

Get_Attributes_All

Tant au niveau instance (gestion d'objets) qu'au niveau classe (gestion de système), ce service retourne une liste de valeurs de paramètres à partir de l'objet.

Reset

Au niveau classe (gestion de système), le service Reset met tous les paramètres à la valeur en usine par défaut.

Apply_Attributes

Tant au niveau instance (gestion d'objets) qu'au niveau classe (gestion de système), ce service fait utiliser activement les valeurs de paramètres dont l'utilisation est en attente.

Get_Attribute_Single

Tant au niveau instance (gestion d'objets) qu'au niveau classe (gestion de système), ce service retourne le contenu de l'attribut spécifié.

Set_Attribute_Single

Tant au niveau instance (gestion d'objets) qu'au niveau classe (gestion de système), ce service modifie le contenu de l'attribut spécifié.

Restore

Au niveau instance (gestion d'objets), ce service restaure les valeurs d'instances de paramètres à partir du stockage non volatil. Au niveau classe (gestion de système), ce service restaure toutes les valeurs de paramètres à partir du stockage non volatil.

Save

Au niveau instance (gestion d'objets), ce service sauvegarde des valeurs d'instances de paramètres dans le stockage non volatil. Au niveau classe (gestion de système), ce service sauvegarde toutes les valeurs de paramètres dans le stockage non volatil.

Get_Member

Au niveau instance (gestion d'objets), ce service retourne le contenu d'un membre sélectionné d'un attribut.

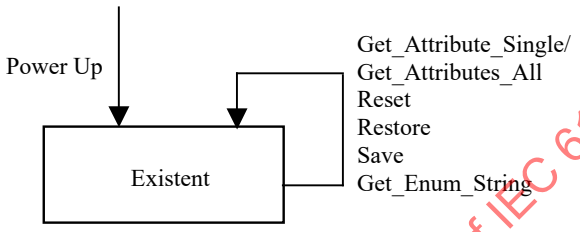
6.2.1.2.7.5 Services spécifiques à un objet

Get_Enum_String

Ce service est utilisé pour lire des chaînes énumérées dans une instance de Parameter.

6.2.1.2.7.6 Diagrammes d'états pour Parameter

Le Tableau 22 est la matrice d'événements d'états pour l'objet Parameter. Le comportement de l'objet Parameter est représenté à la Figure 15.



Anglais	Français
Existent	Existent (Existant)
Get_Attribute_Single/ Get_Attributes_All	Get_Attribute_Single/ Get_Attributes_All
Reset	Réinitialiser
Restore	Restaurer
Save	Sauvegarder
Get_Enum_String	Get_Enum_String
Power Up	Mise sous tension

Figure 15 – Diagramme de transitions d'états pour l'objet Parameter

Tableau 22 – Matrice d'événements d'états pour l'objet Parameter

Evénement	State
	Existent (Existant)
Get_Attribute_Single	Valide/prend en charge la demande et retourne la réponse appropriée.
Set_Attribute_Single	
Get_Attributes_All	
Reset	
Restore	
Save	
Get_Enum_String	

6.2.1.3 Spécification de services de l'ASE Modèle de gestion de FAL

6.2.1.3.1 Services pris en charge

Le paragraphe 6.2.1.3 contient la définition de services qui sont propres à cet ASE. Les services communs définis pour cet ASE sont:

Get_Attributes_All
Set_Attributes_All
Get_Attribute_List
Set_Attribute_List
Reset
Start
Stop
Create
Delete
Multiple_Service_Packet
Apply_Attributes
Get_Attribute_Single
Set_Attribute_Single
Find_Next_Object_Instance
Restore
Save
NOP
Get_Member
Set_Member
Insert_Member
Remove_Member
Group_Sync

Les services spécifiques à un objet définis pour cet ASE sont:

Add_AckData_Path (objet Acknowledge Handler)
Remove_AckData_Path (objet Acknowledge Handler)
Get_Enum_String (objet Parameter)
Symbolic_Translation (objet Message Router)
Flash_LEDs (objet Identity)

6.2.1.3.2 Paramètres communs aux services de la FAL

Les paramètres de service suivants sont communs à tous les services de la FAL.

AREP

Ce paramètre contient des informations suffisantes pour identifier localement l'entité devant être utilisée pour acheminer le service: cette entité pourrait être locale, être l'UCMM ou être une connexion de transports (AR). Ce paramètre peut utiliser un identificateur spécialisé associé à une entité locale, l'identificateur d'un enregistrement de transaction UCMM créé au préalable, ou l'identificateur de transport retourné par le processus d'établissement de connexion et associé à l'AR ("Association Relationship" Relation d'applications) sélectionnée.

Il convient que la primitive "confirmation" utilise la même valeur que celle envoyée dans la primitive "request" correspondante.

Receiver/server local ID (id)

Ce paramètre est créé localement par l'ASE Message Router du nœud de réponse et identifie localement cette invocation du service. Il est utilisé pour associer des réponses de services à des indications. Par conséquent, deux quelconques invocations de services en cours ne peuvent être identifiées par la même valeur d'ID (id) et l'utilisateur de services d'AL doit retourner dans la primitive "response" la valeur qu'il a reçue dans la primitive "indication" antérieure associée.

Path

Dans la demande, ce paramètre spécifie l'APO ("Application Process Object" Objet de processus d'application) de la FAL ou l'élément d'APO de FAL qui est la destination de la demande de service. Le path (chemin) doit contenir plusieurs segments qui peuvent indiquer l'itinéraire réseau jusqu'au nœud contenant l'APO de FAL (dans le cas de plusieurs liaisons), l'identification de l'élément d'APO au sein du nœud-cible (Routing Path) et des informations complémentaires facultatives pour l'APO-cible (Additional Path).

Dans l'indication, ce paramètre doit contenir seulement les segments au-delà du segment de classe logique issu de la demande de service, c'est-à-dire les informations complémentaires facultatives pour l'APO-cible (AdditionalPath).

Voir l'IEC 61158-6-2 pour plus de détails relatifs à la structure de Path et à son contenu.

Port ID

Ce paramètre indique sur quel port de l'appareil l'indication de service est arrivée.

Service status

Ce paramètre donne des informations relatives au résultat d'exécution du service. Il est renvoyé dans toutes les primitives de réponse de service confirmé (+ et -). Il se compose des éléments suivants:

Status code (obligatoire)

Ce paramètre indique si, oui ou non, le service a été traité avec succès. S'il s'est produit une erreur, il indique le type d'erreur. Pour certaines erreurs, une identification plus poussée de l'erreur peut être fournie dans le paramètre Extended Status (c'est-à-dire: statut étendu) ci-dessous. Les codes de statut disponibles sont énumérés en 6.2.1.3.3.

Extended status (conditionnel)

Ce paramètre conditionnel identifie l'erreur qui s'est produite lors du traitement de la demande spécifique à l'objet auquel l'accès est effectué. L'utilisation et la définition réelles de cet Extended status dépendent de la classe d'objets ou du service: chaque classe d'objets définit ses propres valeurs et plages de valeurs de statut étendu (y compris celles qui sont spécifiques à un vendeur). Pour une classe d'objets donnée, le statut étendu est unique pour chaque code de statut. Toutes les valeurs de statut étendu sont réservées, sauf indication contraire dans la définition de l'objet. Lorsqu'il est utilisé, les valeurs présentées dans la primitive "response" sont délivrées inchangées dans la primitive "confirmation".

Ces codes peuvent également être utilisés à d'autres fins telles que l'enregistrement d'événements ou dans des messages heartbeat d'appareil.

6.2.1.3.3 Codes de statut des services FAL

Le Tableau 23 spécifie la plage des codes de statut possibles.

Tableau 23 – Codes de statut

Nom	Description et signification du code de statut
Success (Succès)	Le service a été accompli avec succès par l'objet spécifié.
Connection failure (Echec de connexion)	Un service lié à la connexion a failli le long du chemin de connexion.
Resource unavailable (Ressource non disponible)	Les ressources nécessaires pour que l'objet exécute le service demandé n'étaient pas disponibles.
Invalid parameter value (Valeur de paramètre invalide)	Voir le code de statut 0x20, qui est la valeur préférentielle à utiliser pour cet état.
Path segment error (Erreur de segment de chemin)	L'identificateur de segment de chemin ou la syntaxe de segment n'était pas compris(e) par le nœud de traitement. Le traitement de chemin doit cesser en cas d'erreur de segment de chemin.
Path destination unknown (Destination de chemin inconnue)	Le chemin fait référence à une classe, une instance, un élément de structure d'objet qui n'est pas connu(e) ou n'est pas contenu(e) dans le nœud de traitement. Le traitement de chemin doit cesser en cas d'erreur de destination inconnue.
Partial transfer (Transfert partiel)	Une partie seulement des données prévues a été transférée.
Connection lost (Connexion perdue)	La connexion de messagerie a été perdue.
Service not supported (Service non pris en charge)	Le service demandé n'était pas mis en œuvre ou n'était pas défini pour cette instance de classe ou d'objet.
Invalid attribute value (Valeur d'attribut non valide)	Données d'attribut invalides détectées
Attribute list error (Erreur de liste d'attributs)	Un attribut dans la réponse Get_Attribute_List ou Set_Attribute_List a un statut non nul.
Already in requested mode/state (Déjà dans le mode/état demandé)	L'objet est déjà dans le mode/état demandé par le service.
Object state conflict (Conflit d'état d'objet)	L'objet ne peut pas accomplir le service demandé dans son mode/état actuel.
Object already exists (L'objet existe déjà)	L'instance demandée de l'objet à créer existe déjà.
Attribute not settable (Attribut non réglable)	Une demande de modifier un attribut non modifiable a été reçue.
Privilege violation (Violation de privilège)	Une vérification de permission/privilège a échoué.
Device state conflict (Conflit d'état d'appareil)	Le mode/état courant de l'appareil interdit l'exécution du service demandé.
Response data too large (Données de réponse trop volumineuses)	Les données à émettre dans le tampon de réponses sont plus volumineuses que le tampon de réponses alloué.
Fragmentation of a primitive value (Fragmentation d'une valeur de primitive)	Le service spécifiait une opération qui va fragmenter une valeur de données de primitive, c'est-à-dire, une moitié de type de données REAL.
Not enough data (Pas assez de données)	Le service n'a pas fourni assez de données pour accomplir l'opération spécifiée.
Attribute not supported (Attribut non pris en charge)	L'attribut spécifié dans la demande n'est pas pris en charge.
Too much data (Trop de données)	Le service a fourni plus de données qu'il n'en était prévu.
Object does not exist (L'objet n'existe pas.)	L'objet spécifié n'existe pas dans l'appareil.
Service fragmentation sequence not in progress (Séquence de fragmentation de service pas en cours)	La séquence de fragmentation pour ce service n'est pas active actuellement pour ces données.
No stored attribute data (Aucune donnée d'attribut stockée)	Les données d'attribut de cet objet n'étaient pas sauvegardées préalablement au service demandé.
Store operation failure (Echec de l'opération de stockage)	Les données d'attribut de cet objet n'étaient pas sauvegardées en raison d'une défaillance au cours de la tentative.

Nom	Description et signification du code de statut
Routing failure, request packet too large (Echec d'acheminement, paquet de demande trop volumineux)	Le paquet de demande de service était trop volumineux pour l'émission sur un réseau dans le chemin vers la destination. L'appareil routeur était forcé d'abandonner le service.
Routing failure, response packet too large (Echec d'acheminement, paquet de réponse trop volumineux)	Le paquet de réponse de service était trop volumineux pour l'émission sur un réseau dans le chemin en provenance de la destination. L'appareil routeur était forcé d'abandonner le service.
Missing attribute list entry data (Données d'entrée de liste d'attributs absentes)	Le service n'avait pas fourni un attribut dans la liste d'attributs qui était nécessaire au service pour avoir le comportement demandé.
Invalid attribute value list (Liste de valeurs d'attribut non valide)	Le service retourne la liste d'attributs fournie avec les informations de statut pour les attributs qui n'étaient pas valides.
Embedded service error (Erreur de service intégré)	Un service intégré a donné lieu à une erreur.
Vendor specific error (Erreur spécifique à un vendeur)	Une erreur spécifique à un vendeur s'est produite. Le champ code de statut étendu de la réponse d'erreur définit l'erreur particulière rencontrée. Il convient de n'utiliser ce code de statut général que si aucun des codes de statut présentés dans ce tableau ou dans la définition de classe d'objets ne reflète précisément l'erreur.
Invalid parameter (Paramètre non valide)	Un paramètre associé à la demande n'était pas valide. Ce code est utilisé lorsqu'un paramètre ne satisfait pas aux exigences de la présente spécification et/ou aux exigences définies dans une spécification d'objet d'application.
Write once value or medium already written (Une valeur ou un support inscriptible une seule fois déjà inscrit)	Une tentative est faite d'écrire sur un support inscriptible une seule fois (par exemple, lecteur non réinscriptible (WORM "Write-Once-Read-Many times", PROM "Programmable Read Only Memory") qui a déjà eu une écriture ou de modifier une valeur qui ne peut pas être changée une fois établie
Invalid Response Received (Réponse non valide reçue)	Une réponse non valide est reçue (par exemple, le code de service de réponse ne concorde pas au code de service de demande ou bien le message de réponse est plus court que la taille minimale de réponse prévue). Ce code de statut peut servir pour d'autres causes de réponses non valides.
Buffer overflow (Dépassement de capacité du tampon)	Le message reçu est plus volumineux que ce que le tampon de réception peut gérer. Le message tout entier a été rejeté.
Message format error (Erreur de format de message)	Le format de message reçu n'est pas pris en charge par le serveur.
Key Failure in path (Défaillance de clé dans le chemin)	Le segment de clé qui était inclus comme premier segment dans le chemin ne concorde pas avec le module de destination. Le statut spécifique à un objet doit indiquer quelle partie de la vérification de la clé a échoué.
Path Size Invalid (Taille de chemin non valide)	Soit la taille du chemin qui était envoyé avec la demande de service n'est pas suffisamment grande pour permettre l'acheminement de la demande jusqu'à un objet, soit trop de données d'acheminement étaient incluses.
Unexpected attribute in list (Attribut inattendu dans la liste)	Une tentative a été faite de fixer un attribut qui ne pouvait pas être réglé à ce stade.
Invalid Member ID (Identificateur de membre non valide)	L'ID de membre spécifié dans la demande n'existe pas dans la Classe/l'Instance/l'Attribut spécifié(e).
Member not settable (Membre non réglable)	Une demande de modifier un membre non modifiable a été reçue.
Group 2 only server general failure (Défaillance générale de serveur de groupe 2 seulement)	Ce code de statut peut seulement être rapporté par des serveurs CP 2/3 de Groupe 2 seulement avec un espace de code de 4 Ko ou moins et seulement à la place d'un Service non pris en charge, d'un Attribut non pris en charge et d'un Attribut non réglable.
Unknown Type 15 error (Erreur de type 15 inconnu)	Un traducteur de type 2 à 15 a reçu un code d'exception de type 15 inconnu.
Attribute not gettable (Attribut non récupérable)	Une demande de lire un attribut non lisible a été reçue.
Instance not deletable (Impossible de supprimer l'instance)	L'instance d'objet demandée ne peut pas être supprimée.

Nom	Description et signification du code de statut
Service non pris en charge pour le chemin spécifié	L'objet prend en charge le service, mais pas pour le chemin d'application désigné (par exemple l'attribut). Ce code ne doit être utilisé pour aucun service Set
Reserved for object class specific errors (Réservés pour des erreurs spécifiques de classe d'objets)	Cette plage de codes de statut doit être utilisée pour indiquer des erreurs spécifiques à une classe d'objets. Il convient de n'utiliser cette plage que si aucun des codes de statut présentés dans ce tableau ne reflète précisément l'erreur qui s'est produite.

Toutes les valeurs de statut étendu sont réservées, sauf indication contraire dans la définition de l'objet. Les valeurs spécifiques sont spécifiées le cas échéant dans l'IEC 61158-6-2.

6.2.1.3.4 Get_Attributes_All

6.2.1.3.4.1 Vue d'ensemble du service

Le service Get_Attribute_All retourne le contenu de tous les attributs de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

6.2.1.3.4.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 24.

Tableau 24 – Paramètres du service Get_Attributes_All

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service. Il n'y a pas de paramètres spécifiques pour ce service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Attribute data

Ce paramètre renvoie un flux d'informations incluant les valeurs de chaque attribut que la classe/l'objet définit pour la réponse. Les classes APO qui prennent en charge ce service doivent définir le format de ce paramètre. Aucune fragmentation n'est permise.

S'il est pris en charge, la structure des informations de l'Attribute Data doit être conforme à la structure de la réponse Get_Attribute_All telle que définie par la spécification de la classe d'objets. La spécification de la classe d'objets doit fournir une définition détaillée du format des données devant être envoyées dans les messages de réponse de classes et d'instances.

Les valeurs par défaut doivent être insérées pour tous les attributs non pris en charge qui sont présents dans la matrice de données de la réponse.

Pour les attributs qui spécifient un nombre d'éléments de matrice dans un autre attribut, un nombre zéro et aucun élément de matrice doivent être insérés si la paire d'attributs n'est pas prise en charge.

Pour les attributs qui spécifient un nombre suivi d'une matrice d'entrées, un nombre zéro doit être inséré si l'attribut n'est pas pris en charge.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Response data too large (Données de réponse trop volumineuses)

6.2.1.3.4.3 Procédure de service

Un appareil doit uniquement inclure des données dans la matrice de données de la réponse jusqu'au dernier attribut mis en œuvre.

Si la longueur des données dans la réponse est inférieure à celle indiquée dans la définition de l'objet et si la dernière partie des données de réponse n'inclut aucun attribut complet, les données de réponse sont en statut d'erreur et le client doit les supprimer.

Si la longueur des données dans la réponse est inférieure à celle indiquée dans la définition de l'objet, le client utilise les valeurs par défaut pour les attributs manquants.

Si la longueur des données dans la réponse est supérieure à la quantité attendue, le client ignore toutes les données supplémentaires par rapport à la quantité attendue.

6.2.1.3.5 Set_Attributes_All

6.2.1.3.5.1 Vue d'ensemble du service

Le service Set_Attribute_All modifie le contenu des attributs de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

6.2.1.3.5.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 25.

Tableau 25 – Paramètres du service Set_Attributes_All

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Data	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Attribute data

Ce paramètre spécifie un flux d'informations incluant les valeurs de chaque attribut que la classe/l'objet définit pour la demande. Les classes APO qui prennent en charge ce service doivent définir le format de ce paramètre.

S'il est pris en charge, la structure des informations de l'Attribute Data dans la demande de service doit être conforme à la structure de la demande Set_Attribute_All telle que définie par la spécification de la classe d'objets. La spécification de la classe d'objets doit fournir une définition détaillée du format des données devant être envoyées dans le message de demande.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Device state conflict (Conflit d'état d'appareil)

6.2.1.3.5.3 Procédure de service

Si l'aptitude à modifier un attribut change en fonction de l'état de l'objet, la spécification de l'objet doit spécifier le moment auquel ses attributs peuvent être écrits.

EXEMPLE Ce service pourrait seulement être pris en charge dans un état où tous les attributs sont modifiables.

Les attributs doivent être modifiés seulement si les vérifications des valeurs spécifiques à un attribut (par exemple, vérifications de plage) sont couronnées de succès. Le premier attribut pour lequel la vérification échoue sera spécifié dans le paramètre Additional Code du message de réponse d'erreur. Si une erreur est détectée, la réponse Set_Attribute_All doit retourner un message approprié de réponse d'erreur. Si tous les contrôles de vérification réussissent, les attributs doivent être modifiés et une réponse de succès de Set_Attribute_All doit être retournée.

Si la longueur des données dans le service est inférieure à celle indiquée dans la définition de l'objet et si la dernière partie des données de la demande n'inclut aucun attribut complet, la réponse d'erreur "Not enough data" (0x13) est renvoyée.

Si la longueur des données dans la zone de données du service est inférieure à celle indiquée dans la définition de l'objet, cela signifie que les attributs représentés par les données manquantes ne sont pas pris en charge par le client. Des données manquantes ne doivent entraîner aucune modification des attributs correspondants.

Si la longueur des données dans la demande est supérieure à la quantité attendue, la réponse d'erreur "Too much data" (0x15) est renvoyée.

Si la mise en œuvre d'objet d'un appareil ne prend pas en charge un ou plusieurs des attributs facultatifs contenus dans le format de la demande Set_Attribute_All de cet objet, l'appareil doit alors soit:

- omettre la prise en charge de ce service (si la définition de l'objet le permet);
- rejeter la demande avec une erreur "Attribute Not Supported" (0x14); ou
- accepter la demande fournie; la ou les valeurs contenues dans la demande pour le ou les attributs non mis en œuvre correspondent aux valeurs par défaut définies pour ces attributs. Si l'une des valeurs du ou des attributs non mis en œuvre ne correspond pas à la valeur par défaut définie pour ces attributs, la demande doit être rejetée. Un code d'erreur "Invalid Attribute Value" (0x09) est recommandé dans ce cas.

6.2.1.3.6 Get_Attribute_List

6.2.1.3.6.1 Vue d'ensemble du service

Le service Get_Attribute_List retourne le contenu des attributs récupérables sélectionnés de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

6.2.1.3.6.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 26.

Tableau 26 – Paramètres du service Get_Attribute_List

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Count	M	M(=)		
Attribute List	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Count			M	M(=)
Attribute Data			M	M(=)
Attribute Identifier			M	M(=)
Attribute Status			M	M(=)
Attribute Value			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Attribute count

Ce paramètre indique le nombre d'identificateurs d'attributs dans le paramètre Attribute List en dessous.

Attribute List

Ce paramètre contient la liste des identificateurs d'attributs de haut niveau qui sont demandés de la classe spécifiée ou de l'objet spécifié.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Attribute count

Ce paramètre indique le nombre d'attributs effectivement retournés dans le paramètre Attribute Data en dessous. Ce compte peut être différent si les données demandées ne s'ajustent pas entièrement dans le tampon de réponses (voir la procédure de service en 6.2.1.3.6.3).

Attribute data

Ce paramètre retourne un train d'informations contenant la totalité ou une partie des attributs demandés. Les attributs doivent être récupérés et regroupés dans la séquence spécifiée dans la demande.

Attribute identifier

Ce paramètre contient l'identificateur d'attribut correspondant à la valeur retournée dans Attribute Value.

Attribute status

Ce paramètre indique les informations de statut individuelles pour l'attribut demandé. Il peut soit miroiter l'un des codes de statut de services communs, soit être spécifique à cet attribut d'objet/de classe.

Attribute value

Ce paramètre retourne un train d'informations contenant toutes les données pour les attributs demandés. Les données d'attributs individuels doivent s'ajuster dans leur totalité dans le tampon de réponses (sans fragmentation des attributs individuels). Le service doit accéder à autant d'attributs qu'il peut s'en ajuster dans le tampon de réponses.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Attribute list error (Erreur de liste d'attributs)

6.2.1.3.6.3 Procédure de service

Les attributs doivent être spécifiés en utilisant une liste d'identificateurs d'attributs, qui doit être fournie comme un paramètre de service. S'il n'y a pas assez de place pour les données d'un attribut dans le tampon de réponses, le traitement du service doit s'arrêter et le paramètre Attribute Count doit être mis au nombre d'attributs effectivement regroupés dans le tampon. L'application client (le demandeur) doit vérifier la valeur du paramètre Attribute Count dans la réponse.

NOTE S'il y a lieu, l'application client peut présenter une autre demande de service Get_Attribute_List pour les données d'attribut restantes de sa liste d'origine.

6.2.1.3.7 Set_Attribute_List

6.2.1.3.7.1 Vue d'ensemble du service

Le service Set_Attribute_List met à jour le contenu des attributs sélectionnés de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

6.2.1.3.7.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 27.

Tableau 27 – Paramètres du service Set_Attribute_List

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Count	M	M(=)		
Attribute Data	M	M(=)		
Attribute Identifier	M	M(=)		
Attribute Value	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Count			M	M(=)
Attribute Status List			M	M(=)
Attribute Identifier			M	M(=)
Attribute Status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	

Nom de paramètre	Req	Ind	Rsp	Cnf
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Attribute count

Ce paramètre indique le nombre d'attributs devant être mis à jour et énumérés dans le paramètre Attribute Data en dessous.

Attribute data

Ce paramètre contient un train d'informations contenant la liste effective d'attributs devant être mise à jour et les valeurs de mise à jour correspondantes.

Attribute identifier

Ce paramètre spécifie l'identificateur d'attribut correspondant à l'attribut devant être mis à jour avec le contenu de l'Attribute Value. Le client est incapable de spécifier des sous-éléments d'un attribut.

Attribute value

Ce paramètre contient un train d'informations contenant toutes les données pour les attributs cibles. Les données pour un attribut individuel doivent être placées dans leur totalité dans le tampon de demandes (sans fragmentation).

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Attribute count

Ce paramètre indique le nombre de statuts d'attributs retournés dans le paramètre Attribute Status List en dessous.

Attribute status list

Ce paramètre retourne un train d'informations contenant des informations de statut pour chaque attribut énuméré dans la demande de service.

Attribute identifier

Ce paramètre contient l'identificateur d'attribut correspondant au statut retourné dans Attribute Status.

Attribute status

Ce paramètre indique les informations de statut individuelles pour l'attribut spécifié. Il peut soit miroiter l'un des codes de statut de services communs, soit être spécifique à cet attribut d'objet/de classe.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Attribute list error (Erreur de liste d'attributs)

Device state conflict (Conflit d'état d'appareil)

Missing attribute list entry data (Données d'entrée de liste d'attributs absentes)

6.2.1.3.7.3 Procédure de service

Le service doit être arrêté si le tampon de réponses est incapable d'accepter le statut du prochain attribut.

Les opérations d'établissement individuelles ne doivent pas être interdépendantes (la demande doit être traitée comme une série de demandes d'établissement indépendantes).

6.2.1.3.8 Reset

6.2.1.3.8.1 Vue d'ensemble du service

Le service Reset invoque le service Reset de la classe ou instance d'objet APO spécifiée.

NOTE Typiquement, cela entraînerait une transition vers un état ou mode par défaut.

6.2.1.3.8.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 28.

Tableau 28 – Paramètres du service Reset

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Invalid parameter value (Valeur de paramètre invalide)

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

6.2.1.3.8.3 Procédure de service

Dans le cas où l'exécution de la demande de service Reset placerait l'objet/appareil dans un état qui lui permet de répondre au demandeur, la réponse ne doit pas être retournée tant que le service n'est pas parachevé. Dans le cas où l'exécution du service Reset placerait l'objet/appareil dans un état dans lequel il n'est pas capable de répondre, l'objet/appareil doit répondre avant d'exécuter le service Reset.

6.2.1.3.9 Start

6.2.1.3.9.1 Vue d'ensemble du service

Le service Start invoque le service Start de la classe ou instance d'objet APO spécifiée.

NOTE Typiquement, cela placerait un objet dans un état/mode "running" (en marche).

6.2.1.3.9.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 29.

Tableau 29 – Paramètres du service Start

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Already in requested mode/state (Déjà dans le mode/état demandé)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

6.2.1.3.9.3 Procédure de service

S'il est pris en charge une classe ou instance d'objet, l'effet de ce service doit être tel que documenté dans la spécification d'objet en question.

6.2.1.3.10 Stop

6.2.1.3.10.1 Vue d'ensemble du service

Le service Stop invoque le service Stop de la classe ou instance d'objet APO spécifiée.

NOTE Typiquement, cela placerait un objet dans un état/mode "stopped" (arrêté) ou "idle" (repos).

6.2.1.3.10.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 30.

Tableau 30 – Paramètres du service Stop

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport Identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Already in requested mode/state (Déjà dans le mode/état demandé)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

6.2.1.3.10.3 Procédure de service

L'effet de ce service doit être documenté dans la spécification d'objet individuel.

6.2.1.3.11 Create

6.2.1.3.11.1 Vue d'ensemble du service

Le service Create a pour résultat l'instanciation d'une nouvelle d'instance d'objet au sein de la classe d'objets spécifiée.

6.2.1.3.11.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 31.

Tableau 31 – Paramètres du service Create

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Instance ID			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Instance ID

Ce paramètre obligatoire indique la valeur entière assignée pour identifier l'instance d'objet nouvellement créée.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Already in requested mode/state (Déjà dans le mode/état demandé)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

Missing attribute list entry data (Données d'entrée de liste d'attributs absentes)

Invalid attribute value list (Liste de valeurs d'attribut non valide)

6.2.1.3.11.3 Procédure de service

Si ce service est pris en charge, l'instance d'objet doit être créée et initialisée conformément à la spécification d'objet.

Toute erreur doit se traduire en ce que l'instance d'objet ne soit pas créée.

6.2.1.3.12 Delete

6.2.1.3.12.1 Vue d'ensemble du service

Le service Delete supprime une instance d'objet de la classe d'objets spécifiée.

6.2.1.3.12.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 32.

Tableau 32 – Paramètres du service Delete

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object Specific Data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

6.2.1.3.12.3 Procédure de service

Toutes les ressources doivent être désallouées et retournées au système.

6.2.1.3.13 Get_Attribute_Single

6.2.1.3.13.1 Vue d'ensemble du service

Le service Get_Attribute_Single renvoie les contenus de l'attribut spécifié (ou du sous-attribut spécifié par Bit Index, Indirect Bit Index, Array Index, Indirect Array Index, Structure Member ou Structure Handle) de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

6.2.1.3.13.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 33.

Tableau 33 – Paramètres du service Get_Attribute_Single

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service. Il n'y a pas de paramètres spécifiques pour ce service, sauf comme indiqué ci-dessous.

Path

Ce paramètre contient un segment de chemin spécifiant le numéro d'attribut.

NOTE Pour une description des segments de chemin, voir l'IEC 61158-6-2.

Certains profils CPF 2 prennent en charge les formats de message explicite spécifiques aux liaisons (par exemple, ils ne prennent pas en charge le format de demande Message Router) qui n'autorisent pas la spécification du numéro d'attribut dans le paramètre Path. Pour ces formats, le numéro d'attribut est transmis en tant que paramètre supplémentaire.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Attribute data

Ce paramètre contient les données d'attribut demandées.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Attribute not settable (Attribut non réglable)

Device state conflict (Conflit d'état d'appareil)

Object does not exist (L'objet n'existe pas)

6.2.1.3.13.3 Procédure de service

Aucune procédure de service spécifique.

6.2.1.3.14 Set_Attribute_Single

6.2.1.3.14.1 Vue d'ensemble du service

Le service Set_Attribute_Single modifie les contenus de l'attribut spécifié (ou du sous-attribut spécifié par Bit Index, Indirect Bit Index, Array Index, Indirect Array Index, Structure Member ou Structure Handle) de la classe d'objets spécifiée (attributs de gestion de système d'APO) ou de l'instance d'objet spécifiée (attributs de gestion d'objets APO).

6.2.1.3.14.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 34.

Tableau 34 – Paramètres du service Set_Attribute_Single

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Data	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service. Il n'y a pas de paramètres spécifiques pour ce service.

Path

Ce paramètre contient un segment de chemin spécifiant le numéro d'attribut.

NOTE Pour une description des segments de chemin, voir l'IEC 61158-6-2.

Attribute data

Ce paramètre spécifie la valeur en laquelle l'attribut spécifié doit être modifié. Les données d'attribut doivent être validées avant que la modification n'ait lieu.

Certains profils CPF 2 prennent en charge les formats de message explicite spécifiques aux liaisons (par exemple, ils ne prennent pas en charge le format de demande Message Router) qui n'autorisent pas la spécification du numéro d'attribut dans le paramètre Path. Pour ces formats, le numéro d'attribut est transmis en tant que premier paramètre (octet) des données d'attribut.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Attribute not settable (Attribut non réglable)

Device state conflict (Conflit d'état d'appareil)

Object does not exist (L'objet n'existe pas.)

6.2.1.3.14.3 Procédure de service

Aucune procédure de service spécifique.

6.2.1.3.15 Find_Next_Object_Instance

6.2.1.3.15.1 Vue d'ensemble du service

Le service Find_Next_Object_Instance doit être pris en charge au niveau de la classe d'objets APO uniquement. Il amène la classe d'objets APO spécifiée à rechercher et retourner une liste d'identificateurs d'instance (Instance ID) associés à des instances d'objets existants. Les objets existants sont ceux qui sont actuellement accessibles à partir de la liaison.

6.2.1.3.15.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 35.

Tableau 35 – Paramètres du service Find_Next_Object_Instance

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Receiver/Server Local ID	M	M(=)		
Path size	M	M(=)		
Path	M	M(=)		
Port ID		M		
Maximum Returned Values	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Number of List Members			M	M(=)
Instance ID List			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Path

Ce paramètre inclut un segment de chemin spécifiant un Instance ID, qui sera utilisé pour déterminer le point de départ pour la recherche.

Maximum returned values

Ce paramètre indique le nombre maximal de valeurs d'Instance ID devant être retournées dans le message de réponse.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Number of list members

Ce paramètre contient le nombre effectif des Instance ID retournés dans le message de réponse. Ce nombre d'Instance ID doit être inférieur ou égal au maximum spécifié dans la demande.

Instance ID list

Ce paramètre contient la liste des Instance ID retournés.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

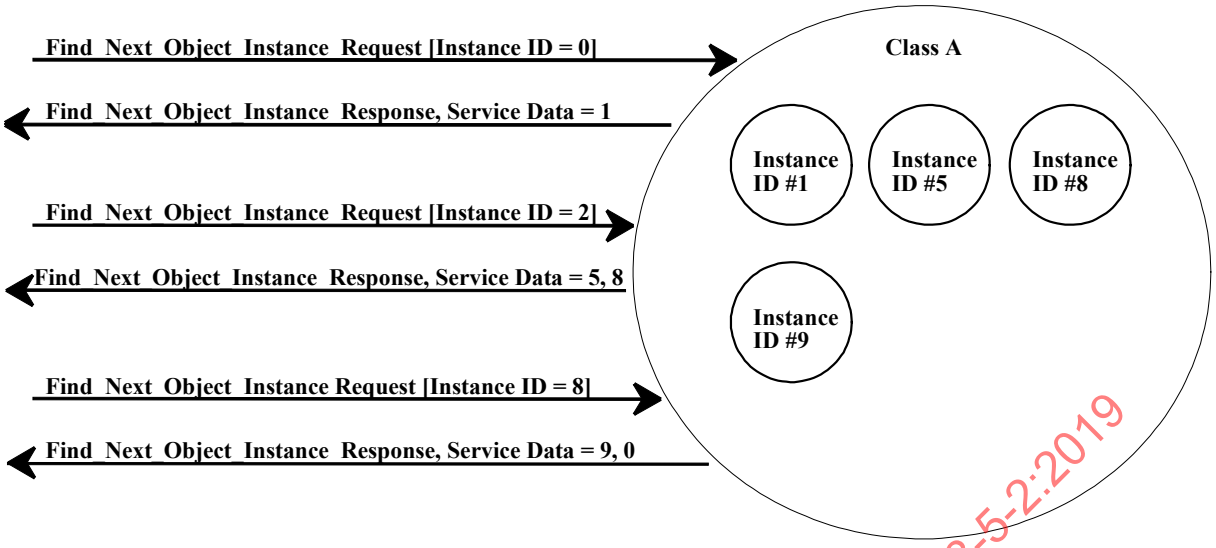
6.2.1.3.15.3 Procédure de service

La classe d'objets doit utiliser la valeur spécifiée dans l'Instance ID du message de demande pour déterminer le point de début de la recherche, comme décrit ci-dessous:

1) Si l'Instance ID dans le message de demande est zéro (0), la classe doit commencer par la plus faible valeur numérique d'Instance ID;
2) Si l'Instance ID dans le message de demande n'est pas zéro (0), la classe doit commencer par le prochain Instance ID dont la valeur numérique est plus grande que celle de l'Instance ID spécifié.
3) Si l'Instance ID dans le message de demande est supérieur ou égal à l'Instance ID ayant la plus grande valeur numérique au sein de la classe, la valeur zéro (0) doit être retournée comme Instance ID.

La classe doit retourner une liste des Instance ID associés à des objets existants, en commençant au point de départ et en continuant dans l'ordre croissant des valeurs d'Instance ID. Elle doit retourner la valeur d'Instance ID zéro (0) pour indiquer que la fin de la liste a été atteinte.

La Figure 16 donne un exemple général de ce service.



Anglais	Français
Find_Next_Object_Instance Request [Instance ID = 0]	Demande de service Find_Next_Object_Instance [Instance ID = 0]
Find_Next_Object_Instance Response, Service Data = 1	Réponse de service Find_Next_Object_Instance, Service Data = 1
Find_Next_Object_Instance Request [Instance ID = 2]	Demande de service Find_Next_Object_Instance [Instance ID = 2]
Find_Next_Object_Instance Response, Service Data = 5, 8	Réponse de service Find_Next_Object_Instance, Service Data = 5, 8
Find_Next_Object_Instance Request [Instance ID = 8]	Demande de service Find_Next_Object_Instance [Instance ID = 8]
Find_Next_Object_Instance Response, Service Data = 9, 0	Réponse de service Find_Next_Object_Instance, Service Data = 9, 0
Class A	Classe A
Instance ID	Instance ID

Figure 16 – Exemple de service Find_Next_Object_Instance

6.2.1.3.16 NOP

6.2.1.3.16.1 Vue d'ensemble du service

Le service NOP amène l'instance d'objet APO de réception à retourner une réponse *No Operation*. L'instance d'objet APO de réception ne doit accomplir aucune autre action interne.

NOTE Ce service peut être utilisé pour tester si, oui ou non, un objet particulier est encore présent et répond sans entraîner de changement d'état.

6.2.1.3.16.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 36.

Tableau 36 – Paramètres du service NOP

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

6.2.1.3.16.3 Procédure de service

Si une instance d'objet à laquelle la demande est remise prend en charge le service NOP, une réponse dont le statut indique un succès doit être retournée. Si l'objet ne prend pas en charge le service NOP, une réponse indiquant qu'une erreur a été détectée doit être retournée.

6.2.1.3.17 Apply_Attributes

6.2.1.3.17.1 Vue d'ensemble du service

Le service Apply_Attributes entraîne que les valeurs d'attribut dont l'utilisation est en cours s'utilisent activement dans la classe ou instance d'objets APO spécifiée.

6.2.1.3.17.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 37.

Tableau 37 – Paramètres du service Apply_Attributes

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

6.2.1.3.17.3 Procédure de service

Les données pour attributs en cours doivent être vérifiées avant de les utiliser.

6.2.1.3.18 Save

6.2.1.3.18.1 Vue d'ensemble du service

Le service Save copie le contenu des attributs d'une classe ou instance d'objets APO en un emplacement accessible par le service Restore.

6.2.1.3.18.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 38.

Tableau 38 – Paramètres du service Save

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

Store operation failure (Echec de l'opération de stockage)

6.2.1.3.18.3 Procédure de service

Le service Save doit rapporter un succès seulement après que la copie a été parachevée et vérifiée.

6.2.1.3.19 Restore

6.2.1.3.19.1 Vue d'ensemble du service

Le service Restore restaure le contenu des attributs d'une classe ou instance d'objets APO à partir d'un emplacement de stockage accessible par le service Save. Les données d'attribut doivent être copiées d'une zone de stockage vers une zone mémoire actuellement active utilisée par la classe ou l'instance d'objet.

6.2.1.3.19.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 39.

Tableau 39 – Paramètres du service Restore

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des paramètres spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Object specific data

Ce paramètre conditionnel, s'il est spécifié, contient des informations spécifiques à une classe/instance d'objet. Si une classe/instance d'objet utilise ce champ, la spécification de la classe/instance d'objet doit détailler le format.

Result(-)

Ce paramètre de type sélection indique que la demande de service a échoué.

Service status

Ce paramètre indique une erreur parmi les choix suivants:

Path segment error (Erreur de segment de chemin)

Path destination unknown (Destination de chemin inconnue)

Connection lost (Connexion perdue)

Service not supported (Service non pris en charge)

Invalid attribute value (Valeur d'attribut non valide)

Object state conflict (Conflit d'état d'objet)

Device state conflict (Conflit d'état d'appareil)

No stored attribute data (Aucune donnée d'attribut stockée)

6.2.1.3.19.3 Procédure de service

Les données d'attribut doivent être vérifiées avant d'accomplir la copie de la "zone de stockage" vers la zone "actuellement active".

Si l'aptitude à modifier un attribut varie en fonction de l'état de l'objet, la spécification d'objet doit donner une description détaillée de la façon dont ce service est effectué. Ce service peut seulement être pris en charge dans un état où tous les attributs sont modifiables. L'objet peut ignorer les données associées à un attribut actuellement non modifiable.

Les attributs doivent être modifiés seulement si les vérifications des valeurs spécifiques à un attribut (par exemple, vérifications de plage) sont couronnées de succès. Le premier attribut échouant à la vérification doit être spécifié dans l'élément Extended Status du paramètre Service Status du message de réponse.

Si une quelconque autre erreur est détectée, la réponse doit être retournée avec un code de statut différent de zéro.

Si tous les contrôles de vérification réussissent, les attributs doivent être modifiés et une réponse de succès de Restore doit être retournée.

6.2.1.3.20 Get_Member

6.2.1.3.20.1 Vue d'ensemble du service

Le service Get_Member renvoie les informations sur le/les membre(s) depuis un attribut.

6.2.1.3.20.2 Primitives de service

Les paramètres de service pour ce service sont montrés dans le Tableau 40.

Tableau 40 – Paramètres du service Get_Member

Nom de paramètre	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Member ID			M	M(=)
Member data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

L'argument contient les paramètres de la demande de service. Il n'y a pas de paramètres spécifiques pour ce service.

Result(+)

Ce paramètre de type sélection indique que la demande de service a réussi.

Service status

Ce paramètre indique un succès.

Member ID

Ce paramètre contient l'identification du membre qui a reçu le service.

Member data

Ce paramètre contient les informations sur le(s) membre(s) issues de l'attribut sélectionné.