IEC 61158-5-2

Edition 5.0 2023-03

INTERNATIONAL STANDARD

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

IEC 61158-5-2:2023-03(en)

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

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

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

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

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 It also gives information on projects, replaced and withdrawn publications.

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

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

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 61158-5-2

Edition 5.0 2023-03

INTERNATIONAL STANDARD

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

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-6569-7

Warning! Make sure that you obtained this publication from an authorized distributor.

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 the IEC 61784-1 series and the IEC 61784-2 series.

IEC 61158-5-2 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

This fifth edition cancels and replaces the fourth edition published in 2019. This edition constitutes a technical revision.

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

a) update of normative and bibliographic references;
b) new STIME, UTIME and NTIME data types in 5.3.1.5;
c) updated list of managements objects in 6.1.3;
d) new attributes and services for the ASE general formal model in 6.2.1.2.1, 6.2.1.3 and 6.5;
e) clarifications, new attributes and services for the Identity ASE in 6.2.1.2.2;
f) clarifications, new attributes and other extensions for the Assembly ASE in 6.2.1.2.3;
g) new attributes and services for the Message Router ASE in 6.2.1.2.4;
h) addition of missing class attributes for the Acknowledge Handler ASE in 6.2.1.2.5;
i) clarifications, new attributes and services for the Time Sync ASE in 6.2.1.2.6;
j) addition of missing class attributes for the Parameter ASE in 6.2.1.2.7;
k) clarifications of service parameters, status codes and procedures in 6.2.1.3;
l) addition of a new service for the Message Router ASE in 6.2.1.3;
m) clarifications and new services for the Connection Manager ASE in 6.2.2;
n) clarifications and new services for the Connection ASE in 6.2.3;
o) removal of obsoleted transport options and related services in 6.3.3;
p) removal of all references to CPF and CPs (material moved to profile documents);
q) miscellaneous editorial corrections.

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

Draft	Report on voting
65C/1203/FDIS	65C/1244/RVD

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

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

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 document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

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

INTRODUCTION

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

The application 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 can 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 document defines in an abstract way the externally visible service provided by the Type 2 fieldbus application layer in terms of:

- an abstract model for defining application resources (objects) capable of being manipulated by users via the use of the FAL service;
- the primitive actions and events of the service;
- the parameters associated with each primitive action and event, and the form which they take; and
- the interrelationship between these actions and events, and their valid sequences.

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

- the FAL user at the boundary between the user and the application layer of the fieldbus reference model; and
- 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 document can 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 the IEC 61784-1 series and the IEC 61784-2 series 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 61158-1:2023, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-2:2023, *Industrial communication networks – Fieldbus specifications – Part 3-2: Data-link layer service definition – Type 2 elements*

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

IEC 61158-4-2:2023, *Industrial communication networks – Fieldbus specifications – Part 4-2: Data-link layer protocol specification – Type 2 elements*

IEC 61158-6-2:2023, *Industrial communication networks – Fieldbus specifications – Part 6-2: Application layer protocol specification – Type 2 elements*

IEC 61588:2021, *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*

IEC 62439-3:2016, *Industrial communication networks – High availability automation networks – Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR)*

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/IEEE 8802-3, *Telecommunications and exchange between information technology systems − Requirements for local and metropolitan area networks − Part 3: Standard for Ethernet*

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 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 8601-1, *Date and time – Representations for information interchange – Part 1: Basic rules*

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*

---

[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.

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*

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 11898-1:2015, *Road vehicles – Controller area network (CAN) – Part 1:Data link layer and physical signalling*

IETF RFC 1759, R. Smith, F. Wright, T. Hastings, S. Zilles, J. Gyllenskog, *Printer MIB,* March 1995, available at https://www.rfc-editor.org/info/rfc1759 [viewed 2022-02-18]

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 https://www.electropedia.org/
- ISO Online browsing platform: available at https://www.iso.org/obp

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.

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.

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

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 can 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 [11] – 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

[11] This definition and several others are based on the legacy IEC 61588:2009 and not the latest IEC 61588:2021.

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 61588:2021, 3.1.12]

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
implementation profile
collection of information and functionality supported by a device, independently of its device type

Note 1 to entry: Unlike device profiles, implementation profiles group subsets of Type 2 technologies and characteristics, providing a high-level description of the capabilities of a device.

3.6.41
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

3.6.42
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 can be used to unambiguously identify the service invocation and differentiate it from other outstanding service invocations.

3.6.43
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.

3.6.44
instance attribute
attribute whose value is unique to an object instance and whose definition is shared by all instances of an object

3.6.45
instantiated
object that has been created in a device

3.6.46
logical device
certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.6.47
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.

3.6.48
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.

3.6.49
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]

3.6.50
member
piece of an attribute that is structured as an element of an array

3.6.51
Message Router
object within a node that distributes messaging requests to appropriate application objects

3.6.52
method
<object> synonym for an operational service which is provided by the server ASE and invoked by a client

3.6.53
module
hardware or logical component of a physical device

3.6.54
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.

3.6.55
network
set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.6.56
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

3.6.57
object specific service
service unique to the object class which defines it

3.6.58
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]

3.6.59
originator
client responsible for establishing a connection path to the target

3.6.60
parent clock
master clock to which a clock is synchronized

[SOURCE: IEC 61588:2009, 3.1.23]

3.6.61
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.6.62
physical device
automation or other network device

3.6.63
point-to-point connection
connection that exists between exactly two application objects

3.6.64
Precision Time Protocol
PTP
protocol defined by IEC 61588:2021

Note 1 to entry: As an adjective, it indicates that the modified noun is specified in or interpreted in the context of IEC 61588:2021.

[SOURCE: IEC 61588:2021, 3.1.48, modified – second sentence changed to a Note, and references to IEEE Std 1588 replaced by IEC 61588:2021]

3.6.65
produce
act of sending data to be received by a consumer

3.6.66
producer
node that is responsible for sending data

3.6.67
property
general term for descriptive information about an object

3.6.68
PTP message
one of the message defined in IEC 61588:2021

[SOURCE: IEC 61588:2021, 3.1.58, modified – NOTE deleted, and reference to IEEE Std 1588 replaced by IEC 61588:2021]

3.6.69
PTP port
logical access point of a clock for PTP communications to the communications network

[SOURCE: IEC 61588:2021, 3.1.61, modified – "PTP instance" changed to "clock"]

3.6.70
resource
processing or information capability of a subsystem

3.6.71
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.

3.6.72
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

3.6.73
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.6.74
synchronized clocks
(to a specified uncertainty) absent relativistic effects, two clocks which have the same epoch and for which measurements of the time of the same single event at an arbitrary instant differ by no more than the specified uncertainty

[SOURCE: IEC 61588:2021, 3.1.75, modified – reworded according to IEC rules]

3.6.75
System Time
absolute time value as defined by Type 2 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 Type 2, System Time is a 64-bit integer value in units of nanoseconds with a value of 0 corresponding to the date 1970-01-01.

3.6.76
target
end-node to which a connection is established

3.6.77
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]

3.6.78
Unconnected Message Manager
UCMM
component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object

3.6.79
unconnected service
messaging service which does not rely on the set up of a connection between devices before allowing information exchanges

3.6.80
vendor ID
identification of each product manufacturer/vendor by a unique number

Note 1 to entry: Vendor IDs are assigned by the ODVA, Inc. organization (see <www.odva.org>).

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
CAN Controller Area Network [ISO 11898-1]
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]
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 can be provided or not 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 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 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 STIME

CLASS: **Data type**

ATTRIBUTES:

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

This IEC 61131-3 type extension is a binary number. This unsigned type has a length of eight octets. The unit of time for this type is 1 ns. STIME represents the time since 1970-01-01 at 00:00:00 (at the 0th meridian). System Time is adjusted for leap seconds, but not local time zones or daylight savings time. This is the same time base defined for Type 2 Time Synchronization System Time (see 6.2.1.2.6.5, a)).

5.3.1.5.3 UTIME

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	UTIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This IEC 61131-3 type extension is a binary number. This unsigned type has a length of eight octets. The unit of time for this type is 1 µs. STIME represents the time since 1970-01-01 at 00:00:00 (at the 0th meridian). System Time is adjusted for leap seconds, but not local time zones or daylight savings time. This is the same time base defined for Type 2 Time Synchronization System Time (see 6.2.1.2.6.5, a)).

5.3.1.5.4 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.

5.3.1.5.5 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.

5.3.1.5.6 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.

5.3.1.5.7 NTIME

CLASS:			Data type
ATTRIBUTES:			
1	Data type Numeric Identifier	=	not used
2	Data type Name	=	NTIME
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 ns.

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), and 0xDA (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 and IEC 61158-1, 9.8 are not used; instead they are specified in relevant subclauses.

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)

AR ASE:

- Unconnected Message Manager (queued messaging services on a single link)

Depending on the network communication profile, every node shall contain one or several instances of the following ASE object classes in its application layer.

Connection Manager ASE:

- Connection Manager (establishment and maintenance of connections)

and/or Connection ASE:

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

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 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 or other intermediate layers:

- 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 11898-1:2015 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),
- LLDP Management object (administrative information for the LLDP protocol),
- LLDP Data Table object (record of all adjacent LLDP devices that are currently active).

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:

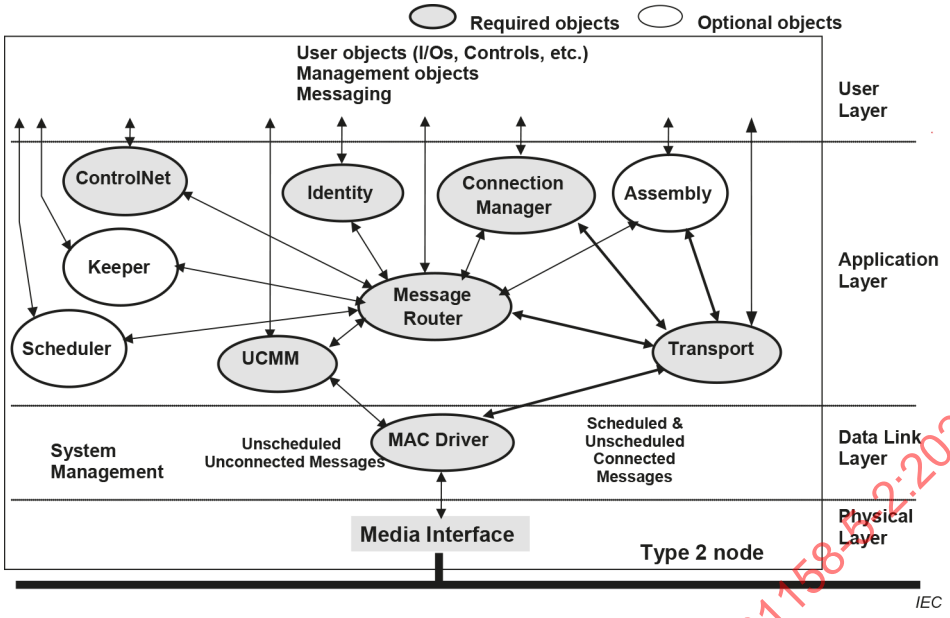

Figure 1 – Overview of ASEs and object classes

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.

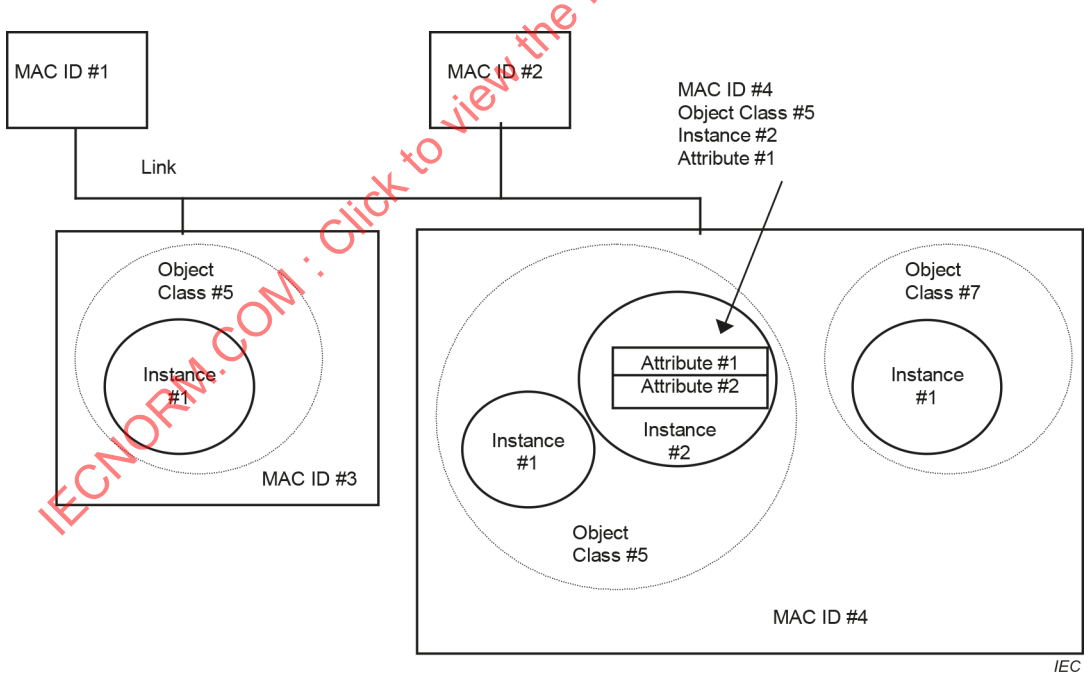

Figure 2 – Addressing format using MAC, class, instance and attribute IDs

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 can 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
TIME
STIME
UTIME
ITIME
FTIME
LTIME
NTIME
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 document.

NOTE 2 IEC 61131-3:2003, 1.5.1, defines the requirements which are met by programmable controller systems claiming compliance with 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 document. 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
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: >, >=, =, <=, <, <>

IEC 61131-3:2003 Table/feature	Feature description
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 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 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 document 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	STIME data type UTIME data type ITIME data type FTIME data type LTIME data type NTIME 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

6.1.6 Diagnostic connection points

Many ASEs have attributes that are useful as diagnostic information. In order to simplify referencing these attributes as a group, each ASE may define one or more structures as diagnostic connection points (see ASE definitions for connection point definitions). The content chosen to be a part of these structures is generally limited to items that:

- when the value changes, it indicates that there can be a problem with the device that needs attention, or that a previous problem has cleared;
- indicate the current device resource loading (e.g. number of TCP connections, I/O connections, packet rates).

Values that are generally static like device settings, capabilities attributes, will not be part of the structure.

Bit alignment of 64-bit, 32-bit, 16-bit and 8-bit values is maintained and overall size is padded for 64-bit alignment if necessary. This is required so that when multiple connection points from various objects are aggregated in a Diagnostic Assembly, they will collectively maintain for each data item the 8, 16, 32 and 64-bit alignment requirements.

The padding between connection point members shall be represented as null members in the connection point member list. See the description of the Get_Connection_Point_Member_List service in 6.2.1.3.31.

In order to allow clients to understand potential future enhancements/changes to the content of these structures, the following rules for how the structures can be modified apply.

- Devices shall not alter the connection point structure (i.e. no attributes may be added, deleted or substituted) that is defined by each object.
- Products implementing the connection point shall implement all member attributes (i.e. no blanks or placeholders for unimplemented attributes).
- Existing connection points shall only be extended with new members. It is not permissible to modify, reorder or delete existing members.
- If extended by the object definition, the original connection point structure will be maintained in the extended connection point definition.

  NOTE This allows clients who have not been updated for the extended structure to still understand and use the data associated with the prior version of the connection point, thereby providing for backwards compatibility.

- If the connection point is modified in such a way that it inserts new members, reorders existing members, removes existing members, or otherwise changes the meaning of existing members, then a new connection point ID is assigned to the new structure.

If a diagnostic connection point structure is extended, updated clients accessing this structure in devices shall expect to receive either structure depending on which version the device implements. If an older client encounters a device that implements a larger structure, it shall only process the smaller amount that it understands.

It is recommended that all tools use the attribute name when presenting these values to a user, so there is consistent naming across all products and vendors.

Devices that implement diagnostic connection points shall also implement the Get_Connection_Point_Member_List service (see 6.2.1.3.31).

Devices that implement diagnostic connection points shall also allow access to the connection point using the Get_Attribute_Single service directed to the connection point (PATH: Class, Instance, Connection Point). Diagnostic Connection Point access is Get only regardless of the access rules of the individual members.

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 can 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 can 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	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
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
200	(*)	Attribute:	Max Instance (long)

201	(*)	Attribute:	Number of Instances (long)

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
29	(o)	Mgt Service:	Get_Connection_Point_Member_List

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
29	(o)	Ops Service:	Get_Connection_Point_Member_List

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.

Nine predefined Class Attribute IDs are reserved for class object definition. The nine predefined/reserved class attributes have the definitions listed below. Because these attributes are reserved, class Attribute ID numbers 1 through 7, 200 and 201 are always reserved. Therefore, if a class attribute is added to an object specification, it shall start with AttributeID #8 and be different from 200 or 201.

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 (short/long)

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.

This attribute is optional. If the device implements this attribute, it shall implement the long version if it supports instances greater than 65 535, else it shall implement the short version.

Number of Instances (short/long)

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

This attribute is optional. If the device implements this attribute, it shall implement the long version if it supports instances greater than 65 535, else it shall implement the short version.

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 General Status code “Attribute Not Supported” 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 or other logical elements 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.

Get_Connection_Point_Member_List

The Get_Connection_Point_Member_List service returns EPATHs and associated sizes of the value for member attributes defining the structure of a connection point.

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	(*)	Attribute:	Revision = 2
2	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
200	(*)	Attribute:	Max Instance (long)
201	(*)	Attribute:	Number of Instances (long)

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	(*)	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
21	(o)	Attribute	Catalog Number
22	(o)	Attribute	Manufacture Date
23	(*)	Attribute	Type9 Identity Info
24	(*)	Attribute	Type9 Status
25	(o)	Attribute	Implementation Profiles
26	(*)	Attribute	IEC 61131-9 Identity Info 1
27	(*)	Attribute	IEC 61131-9 Identity Info 2
28	(*)	Attribute	IEC 61131-9 Identity Info 3
29	(*)	Attribute	IEC 61131-9 Identity Info 4
30	(*)	Attribute:	Supported Language List 2
31	(o)	Attribute:	Vendor Name
32	(o)	Attribute:	Vendor URI
33	(o)	Attribute:	Configuration Counter
34	(o)	Attribute:	Configuration Date
35	(o)	Attribute:	Manufacture Serial Number
36	(o)	Attribute:	Device Manual
37	(*)	Attribute:	Hardware Revision
37.1	(m)	Attribute:	Major Revision
37.2	(m)	Attribute:	Minor Revision
39	(o)	Attribute	Product Identity Certificate

SYSTEM MANAGEMENT SERVICES:

1	(o)	Mgt Service:	Get_Attributes_All
3	(o)	Mgt Service:	Get_Attribute_List
4	(o)	Mgt Service:	Set_Attribute_List
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
3	(*)	Ops Service:	Get_Attribute_List
4	(*)	Ops Service:	Set_Attribute_List
5	(*)	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
76	(*)	Ops Service:	Certificate_Challenge

6.2.1.2.2.2 System management attributes (class attributes)

Revision

If the value is 1, then this attribute is **optional**. If the value is greater than 1, then this attribute is **mandatory**.

Max instance (short/long)

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.

Specific Vendor IDs are specified for devices implementing a different network technology and connected through a Type 2 translator device. Because this Vendor ID is assigned to a network technology and not vendor specific, it cannot be used to identify the company that defines any vendor specific behavior for a translated device. Instead, the Vendor ID of the translation device shall be used to identify the vendor who defines vendor-specific behavior for the set of its translated devices.

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 basic set of data, and contain the same basic 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.

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

Revision

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

The Major and Minor Revision are typically displayed as "major.minor". Minor revisions shall be 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 1 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.

Typically this will be due to the device being an end point of an I/O connection to an ownable resource within the device, such as output data. A resource may also be owned through explicit messaging. The existence of an I/O connection alone does not indicate ownership, such as could be present through a Type 2 router.

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 2 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 3 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 4 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 could be unable to communicate in the Major Unrecoverable Fault state. Therefore, it could be unable 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 5 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 6 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 7 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.

A product may automatically modify the Configuration Consistency Value whenever any non-volatile attribute is altered. A client node has the possibility to compare this value to a value within its own memory prior to system operation.

NOTE 8 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

If the Revision value is 1, then this attribute is **optional**. If the value is greater than 1, then this attribute shall not be used.

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.

The following error responses are defined for a Set_Attribute_Single service of the Semaphore attribute:

- Resource Unavailable: The Client Electronic Key portion of the Semaphore attribute was non-zero and did not match the Client Electronic Key portion of the Set_Attribute_Single service data. This error response indicates that the Semaphore has been previously set by a different client process and is not available at the present time.
- Invalid Attribute Value: The value specified for the timer portion of the attribute value was less than 100 ms.
- Device State Conflict: The device is currently in a state that does not allow the setting of the Semaphore attribute. This is a device specific behavior and shall be documented by the vendor.

This volatile 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.

Catalog number

Textual catalog or model number. This shall be the text string used to order the product.

A device could be unable to return the complete catalog number for a number of reasons, for example mechanical options, hardware options or field installable options. Even if the complete catalog number cannot be provided, it is highly recommended that a device returns as much of the catalog number for the product as practical/possible.

One approach is to provide a portion of the catalog number that represents a family of products, possibly using wildcard characters to represent options.

EXAMPLE 1438-BAC7xx" where "xx" represents variants in the catalog number supported by this product code.

NOTE 9 The logical view to the network (described in the "Product Code" attribute) does not include the Catalog Number.

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

Manufacture date

Date the product was manufactured.

A recommended display format for date according to ISO 8601-1 is "yyyy-mm-dd", where "yyyy" is the year, "mm" is the month, and "dd" is the day of the month.

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

Type9 identity info, Type9 status

These attributes are reserved for Type 9 devices. Otherwise they shall not be used.

Implementation profiles

The Implementation profiles attribute is used to identify the implementation profiles supported by a particular product. Implementation profiles define the minimum requirements that a product shall implement, as well as common options.

NOTE 10 The list of public implementation profiles is managed by the ODVA, Inc. organization. Definition of the implementation profiles is outside the scope of this document.

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

IEC 61131-9 identity info 1, info 2, info 3, info 4

These attributes are reserved for IEC 61131-9 devices. Otherwise they shall not be used.

Supported language list 2

If the Revision value is greater than 1, then this attribute is **optional**. If the Revision value is 1, then this attribute shall not be used.

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

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

Vendor Name

Name of the company that manufactured the device.

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

Vendor URI

This attribute provides a globally unique identifier for the manufacturer/vendor of the device (corresponding to the Vendor ID attribute). This identifier can be a fully qualified domain name or a UUID. This value, used in conjunction with the Serial Number attribute, forms a unique identifier for each device on any Type 2 network.

If this attribute is not supported, a client or gateway shall report a default value based on its Vendor ID.

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

Configuration Counter

Number of times the device configuration has changed.

A product may automatically modify the Configuration Counter whenever any non-volatile attribute is altered. A client node has the possibility to compare this value to a value within its own memory prior to system operation. The client node's behavior, upon detection of a mismatch, is vendor specific. The Configuration Counter shall use an incrementing count mechanism and wrap around from its maximum value to zero. A Type 1 Reset shall not set this counter to zero, instead it shall be incremented from its current value.

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

Configuration Date

This attribute contains the date and time of the last configuration change. A change to the Configuration Counter or the Configuration Consistency Value attributes shall accompany a new value stored in Configuration Date. If no time source is available when configuration has changed, a value of zero shall be recorded and subsequently reported.

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

Manufacture Serial Number

Alphanumeric character sequence used to identify the asset.

The Manufacture Serial Number, when used in conjunction with the Vendor ID, shall be unique for each device, and is intended to be used for asset tracking, determining effectivity of any product warranty, and for determining if a product notice (i.e. recall) applies to a particular device.

This attribute may contain characters that duplicate those of the Serial Number attribute or may be a new code assigned by the vendor that contains alphanumeric characters.

If the product cannot determine the Manufacture Serial Number, the value of this attribute shall be an empty string (length 0).

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

Device Manual

Pointer to a manual for the device. This attribute shall contain either an application path to an instance of the File Object containing the device's manual, or a URL where the device's manual can be obtained online.

The first member of the structure indicates the data type of the rest of the attribute.

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

Hardware Revision

If the item the instance represents has hardware, then this attribute is **optional**, else this attribute shall not be used.

This attribute identifies the vendor assigned revision of the hardware that this instance of the Identity object represents.

The value of this attribute can change without a change to the Revision attribute of this instance.

The Major and Minor Revision are typically displayed as major.minor. Minor revisions shall be displayed as three digits with leading zeros as necessary.

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

Major revision

A Major Revision change is required when functional changes are made to the hardware. For example, introducing a new flash memory component that requires an updated firmware driver for programming can warrant a change in the Major Revision.

Minor revision

A Minor Revision change is required when no functional changes result from the hardware changes. A drop-in replacement component change that requires no firmware update can be an example of a Minor Revision.

Product Identity Certificate

This attribute is reserved for security related information.

NOTE 11 Type 2 security is outside the scope of this document.

This non-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, unless otherwise specified in the network communication profile, this service is **mandatory**.

Get_Attribute_List

At the instance level, this service is **mandatory** if the Revision value is greater than 1 and attributes not included in the Get_Attributes_All response are implemented, else it is **optional**.

Set_Attribute_List

At the instance level, this service is **mandatory** if the Revision value is greater than 1 and settable attributes not included in the Get_Attributes_All response are implemented, else it is **optional**.

Reset

At the instance level, this service is **mandatory** for instance 1, else it is **optional** for instances greater than 1.

If directed to instance 0 or 1, this service invokes the Reset service for the entire device. If directed to an instance greater than 1, the scope of the Reset service is vendor-specific.

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:

a) determine if it can provide the type of reset requested, and return the appropriate error if it cannot;

b) respond to the request;

c) perform the type of reset requested.

If the reset is directed to instance 0 or 1, 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 General Status code “Device State Conflict”. The device shall resume accepting explicit message service requests after the reset has completed.

If the instance targeted does not support the Reset service, then a General Status code "Service not supported for specified path" shall be returned.

A client that sends a Reset service to the device should wait at least 3 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**.

6.2.1.2.2.5 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).

Certificate_Challenge

This service is reserved for security. It is **mandatory** if attribute Product Identity Certificate is implemented, else it shall not be used.

NOTE Type 2 security is outside the scope of this document.

6.2.1.2.2.6 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 could be unable to communicate in the Major Unrecoverable Fault state. Therefore, it could be unable 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.

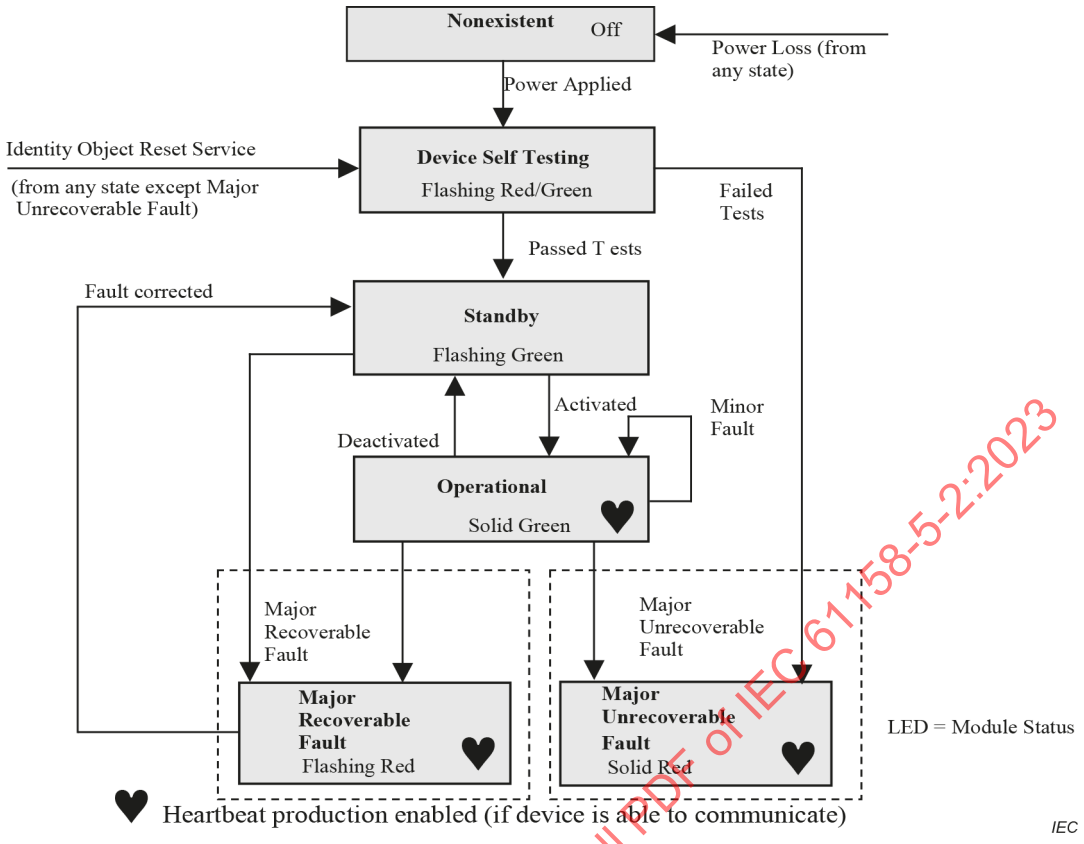

Figure 3 – Identity object state transition diagram

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 (whether or not communications channels are already 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 (whether or not communication channels are still 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.

Devices that implement Type 2 security are required to allow certain authenticated requests, including disruptive requests, while in Explicit and/or Implicit Protection mode.

NOTE Type 2 security is outside the scope of this document.

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.

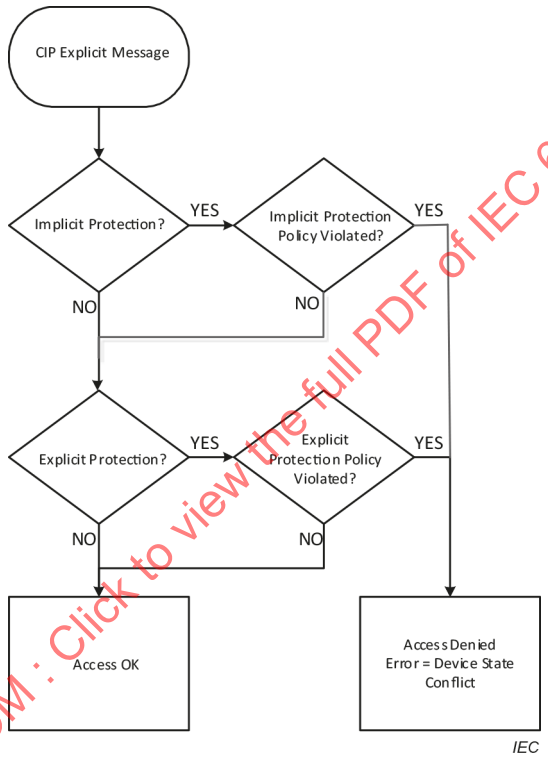

Figure 4 – Explicit and Implicit Setting interaction

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 can be represented by an attribute of one of the device internal objects. Communicating multiple pieces of data (attributes) across a single connection can 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 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, static or variable, and differ by their behavior.

- Dynamic Assemblies use assemblies member lists created and managed by the user of the device. The member list may be altered by the manager of the Dynamic Assembly by adding or deleting members. Devices that support dynamic assembly(ies) with Member List Signature attribute shall support the Electronic Key segment Format 5 (includes serial number).
- Static Assemblies use member lists defined by the device profile or by the product implementers (device manufacturer). The Instance number, number of members, and member list shall be fixed. They shall not be modified.
- Variable Assemblies have a predefined member list for all device types, with limited variability based on device features. The member list is not settable, but may change based on device options, features and port usage assignments made by the user. Devices that support variable assembly(ies) shall support the Electronic Key segment Format 5 (includes serial number).

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. Variable assemblies shall be assigned instance IDs in the Predefined Assemblies range; there is no vendor specific range for this assembly type.

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 = 3
2	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
200	(*)	Attribute:	Max Instance (long)
201	(*)	Attribute:	Number of Instances (long)

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
5	(*)	Attribute:	Member List Signature
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	(c)	Constraint:	Assembly type = STATIC | DYNAMIC
25.1	(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 or Variable Assembly.

Number of members in "Member List" attribute below.

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

Member list

This instance attribute is **optional** for a Static Assembly, **mandatory** for a Dynamic or Variable 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 non-volatile instance attribute has an access rule of Get only for a Static or Variable 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. The volatile or non-volatile status of this instance attribute is defined by each item in the Member List.

Size

Number of octets in "Data" attribute.

This instance attribute has an access rule of Get only. It is typically non-volatile, but the Size can change if a member has a variable size (for example a STRING).

Member list signature

This instance attribute is **optional** for a Static Assembly, **mandatory** for a Variable Assembly, and **optional** but recommended for a Dynamic Assembly.

The Member List Signature is used to indicate that the Member List attribute has changed.

This non-volatile instance attribute has an access rule of Get only for a Static or Variable Assembly, of Set for a Dynamic Assembly.

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

This value is maintained by the device to indicate that the Member List attribute has changed and that the content of the Data attribute can now be different than it was previously. It shall be a counter value that increments by one, with each change to the Member List attribute and wraps around to 0. This value shall be retained through power cycles.

When the "Member List Signature" attribute is implemented, the "Number of members in list" and "Member List" attributes shall also be implemented. When implemented, the "Member List Signature" attribute shall be the first member in the "Member List" attribute so that the "Member List Signature" value is the first value in the assembly Data attribute.

Clients shall utilize attributes "Member List", "Data" and "Member List Signature" together to assure that the client understands the member list content, using the following steps:

1) Open Class 3 explicit messaging connection
2) Read "Member List Signature" attribute
3) Read "Member List" attribute
4) Loop Reading "Data" Attribute. If the "Member List Signature" read in step 2) doesn't match the "Member List Signature" in the Data attribute, go to step 2), else use data

If using unconnected messaging, the Electronic Key Segment type 5 (includes serial number) should be included to verify the device has not been replaced.

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

Create

At the class level (system management), the Create service instantiates a Dynamic Assembly instance. Response contains the instance number.

The Create service supports an optional request parameter "Member List Signature Flag", used to indicate if the newly created instance will support the Member List Signature attribute.

If the Member List Signature instance attribute is not supported and the flag value is true (“Include Member List Signature”), an error shall be returned indicating “Invalid Parameter”.

When the flag value is true ("include Member List Signature"), the newly created instance:

- supports the Member List Signature attribute;
- defaults its Member List Signature attribute value to 0;
- prepends the Member List Signature attribute to the instance's Member List attribute;
- sets the Number of Members in List attribute to 1.

When the flag value is false ("omit Member List Signature"), the newly created instance:

- shall not support the Member List Signature attribute;
- has an empty Member List attribute;
- sets the Number of Members in List attribute to 0.

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 **conditional** for a Static Assembly, else it it **mandatory.** For a Static Assembly, this service is **mandatory** if the Max Instance system management attribute or the Member List object management 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 Set_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.

When a Set_Attribute_Single, Set_Member, Insert_Member or Remove_Member service is targeted at the Member List attribute of a Dynamic instance that supports the Member List Signature attribute, the Member List Signature attribute of that instance is incremented by one when the service is successful.

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.

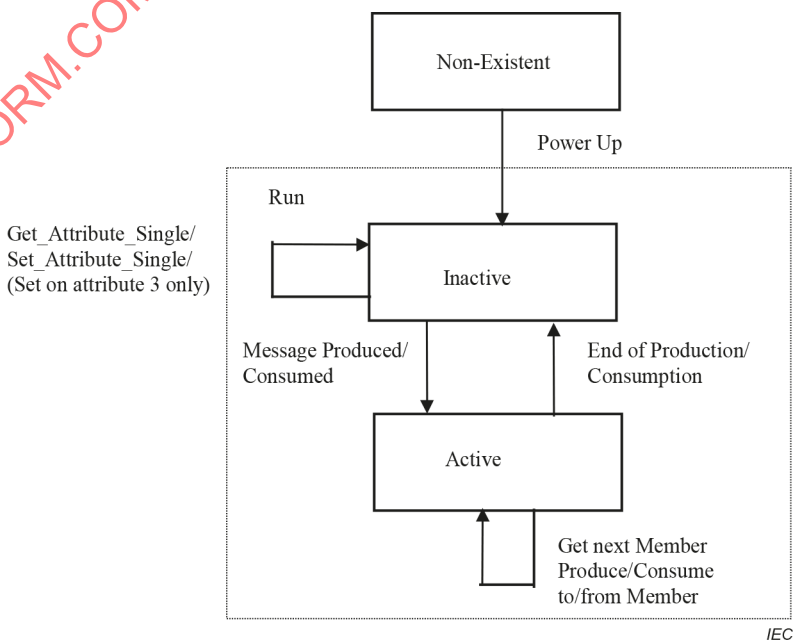

Figure 5 – Static Assembly state transition diagram

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 instance does not exist.	Validate/service the request. Return response	Validate/service the request. Return response
Set_Attribute_Single	Error: Object instance does not exist.	Validate/service the request. Return response	Error: Object state conflict
Message produced/ consumed	Error: Object instance 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 instance 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
Size	Not available	Read only	Read only
Member List Signature	Not available	Read only	Read only

Dynamic Assembly state machine

Figure 6, Table 10 and Table 11 illustrate the behavior of Dynamic Assembly objects.

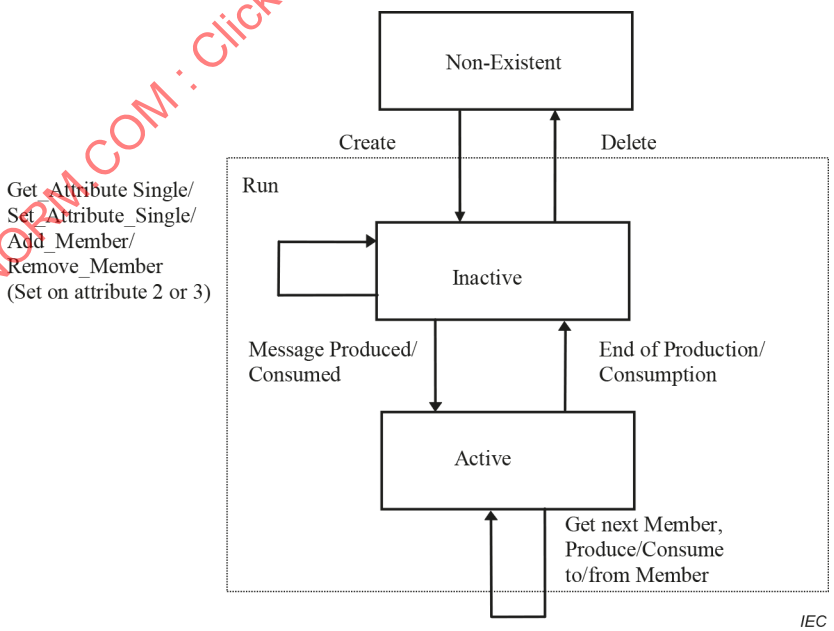

Figure 6 – Dynamic Assembly state transition diagram

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 instance does not exist.	Release all associated resources. Transition to Non–Existent	Error: Object state conflict
Get_Attribute_Single	Error: Object instance does not exist.	Validate/service the request. Return response	Validate/service the request. Return response
Set_Attribute_Single	Error: Object instance does not exist.	Validate/service the request. Return response	Error: Object state conflict
Insert_Member	Error: Object instance does not exist.	Validate/service the request. Return response	Error: Object state conflict
Remove_Member	Error: Object instance does not exist.	Validate/service the request. Return response	Error: Object state conflict
Message produced/ consumed	Error: Object instance 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 instance 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
Size	Not available	Read only	Read only
Member List Signature	Not available	Read only	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).

Variable Assembly state machine

Variable assemblies shall implement the attributes Number of Members in List, Member List and Member List Signature. The Member List Signature shall be the first member specified in the Member List. Because Variable assembly content can change independent of data production/consumption, it shall be the responsibility of unconnected and Class 3 connected clients interacting with Variable Assemblies to verify that the Member List has not changed before attempting to use the contents from the Data attribute. If a Variable Assembly is used by a Class 0/1 client, the content shall not be changed as indicated in the SEM.

Figure 7, Table 12 and Table 13 illustrate the behavior of Variable Assembly objects.

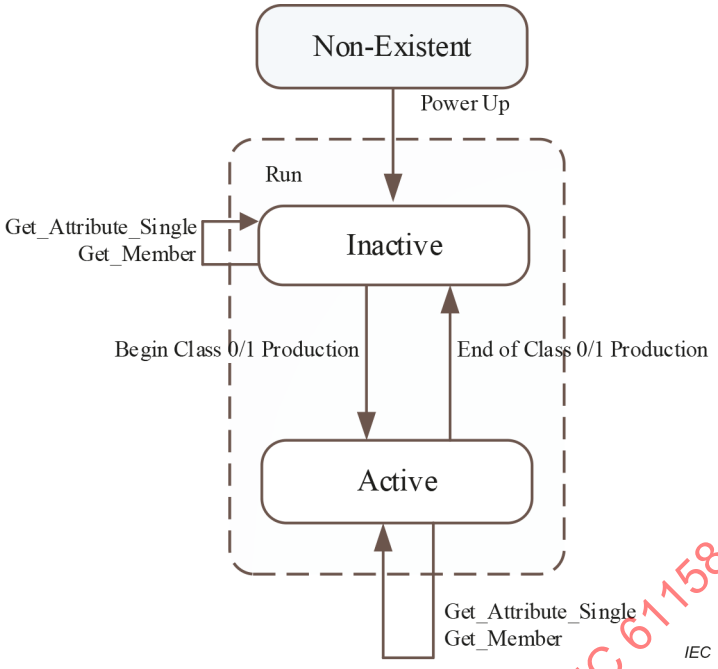

Figure 7 – Variable Assembly state transition diagram

Table 12 – Variable Assembly state event matrix

Event	Variable Assembly object state		
	Non-existent	Inactive	Active
Power Up	Transition to Inactive	Not applicable	Not applicable
Get_Attribute_Single	Not applicable	Validate/service the request Return response	Validate/service the request Return response
Change to Member List	Not applicable	Update Member List Attribute and Member List Signature Attribute as necessary.	Not permitted
Begin Class 0/1 message production	Not applicable	Begin producing from each member in list, Transition to Active	Not applicable
End of Class 0/1 message production	Not applicable	Not applicable	Transition to Inactive

When the Member List is changed (the object state shall be Inactive), the assembly shall update the contents of the Data attribute accordingly.

Table 13 – Variable Assembly instance attribute access

Attribute	Variable 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
Member List Signature	Not available	Read only	Read only
Data	Not available	Read only	Read only

6.2.1.2.3.6 Specific requirements for a connection to the Assembly object

Connection points

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.

Predefined Diagnostic Assembly instances

A range of assembly instances are reserved for predefined diagnostic assembly instances. These instances are defined for optional use by products to provide access at a known location, regardless of device type or network type. There is no vendor specific range for predefined assembly types. The predefined assemblies can be static, dynamic or variable.

Standard Network Diagnostic Assembly

The Standard Network Diagnostic Assembly is a Variable assembly that contains information related to the diagnostic status of the network interface of the device. A client will read this structure and present it to an application or user for interpretation of network health, in terms of for example connection status, device loading.

This assembly will have different members based on factors such as the number of ports and specific device features, some of which may be user assignable (such as port usage). Therefore, this structure can change if a user reconfigures the device and the new configuration results in a change in features/options/port usage. Such a change is only permitted when the state of this instance is Inactive.

The content chosen for these assemblies comes from connection point definitions that are defined for various object classes (see specific object classes for the definitions). The members of this structure are, by nature, network-specific, and are defined in the specified object definitions. The object class structures permitted in this assembly are specified in IEC 61158-6-2, 4.1.8.4.1.4. If the device supports an object which has a Diagnostic Connection Point shown in IEC 61158-6-2, 4.1.8.4.1.4, then it shall support that Connection Point.

The Standard Network Diagnostic Assembly shall be accessible via Explicit (connected and unconnected) and Implicit messaging. However, if the structure size exceeds the capacity of the Forward_Open based transport or unconnected messaging, then the Large_Forward_Open shall be supported to access the Data attribute and there will be no unconnected support.

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 can 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
2	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list

6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
200	(*)	Attribute:	Max Instance (long)
201	(*)	Attribute:	Number of Instances (long)
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
3	(o)	Attribute:	Number active
4	(o)	Attribute:	Active 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
10	(o)	Ops Service:	Multiple_Service_Packet
14	(*)	Ops Service:	Get_Attribute_Single
OBJECT SPECIFIC SERVICES:			
75	(o)	Mgt Service:	Symbolic_Translation
76	(o)	Ops Service:	Send_Receive_Fragment

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.

Number active

Number of connections currently allocated to system communication.

This instance attribute has an access rule of Get only.

Active connections

List of the connection IDs of the currently active connections.

Array of system connection IDs

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.

Send_Receive_Fragment

This service accumulates multiple fragmented packets into a large request, invokes the embedded service and returns the fragmented response.

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 14 shall also be supported.

Table 14 – 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
2	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
200	(*)	Attribute:	Max Instance (long)
201	(*)	Attribute:	Number of Instances (long)
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 15 is the state event matrix for the Acknowledge Handler object associated with a Change of State connection. Table 16 is the state event matrix for the producing I/O application object.

Table 15 – 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

Event	Acknowledge Handler object state		
	Non-existent	**Inactive**	**Active**
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.
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 16 – 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
Acknowledge active	Set Ack Active Flag	1) Transition to Running with Acknowledgement 2) Set Ack Active Flag	Ignore event	Set Ack_Active Flag

Event	Producing I/O application object state			
	Not running	Running	Running with acknowledgment	Prohibited
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 can 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.

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

b) 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).

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

d) 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.

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

Figure 8 shows the typical relationship of timers within the Acknowledge Handler object. An example of the typical timing relationships is shown in Figure 9.

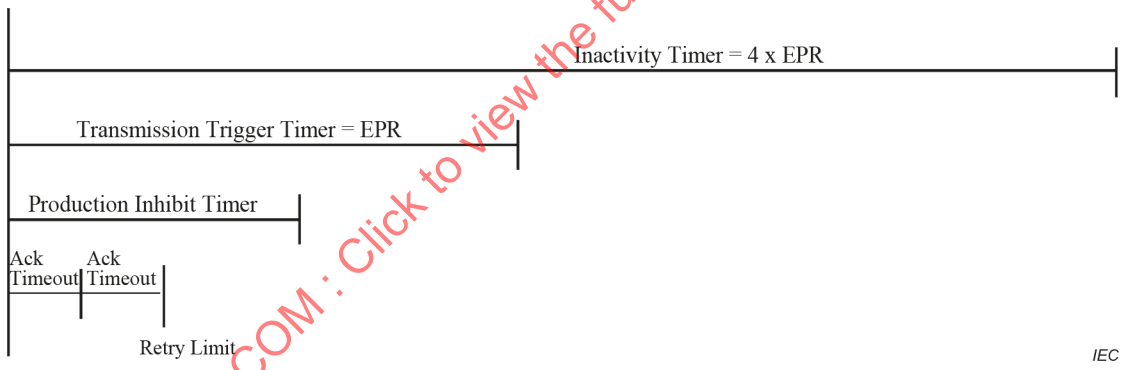

Figure 8 – Typical timing relationships for acknowledged data production

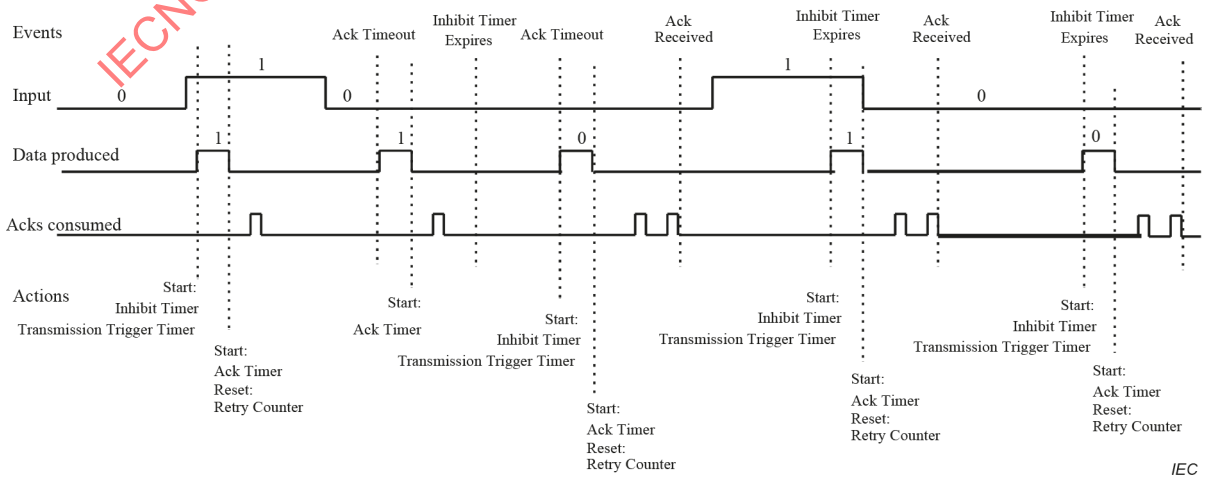

Figure 9 – Example of a COS system with two acking devices

The following diagrams illustrate the message flow in a Change of State connection for both single (see Figure 10) and multi-consumer configurations (see Figure 11).

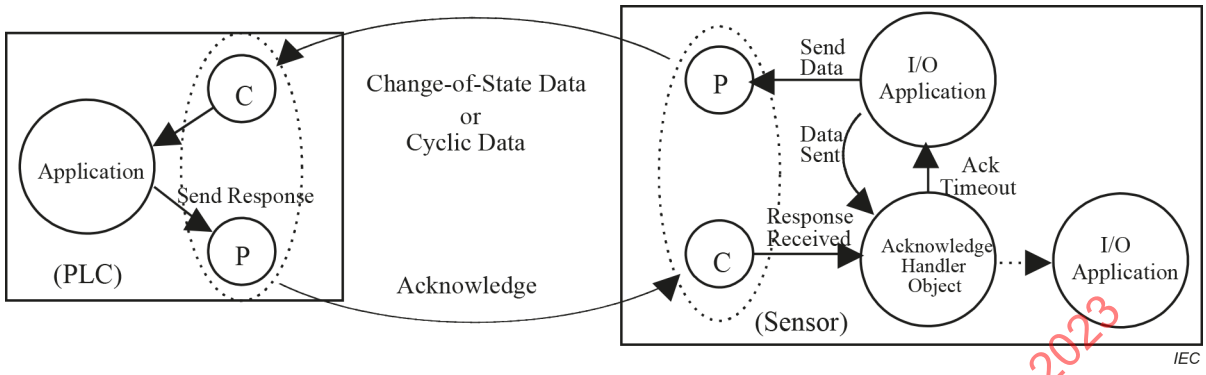

Figure 10 – Message flow in COS connection – one Connection object, one consumer

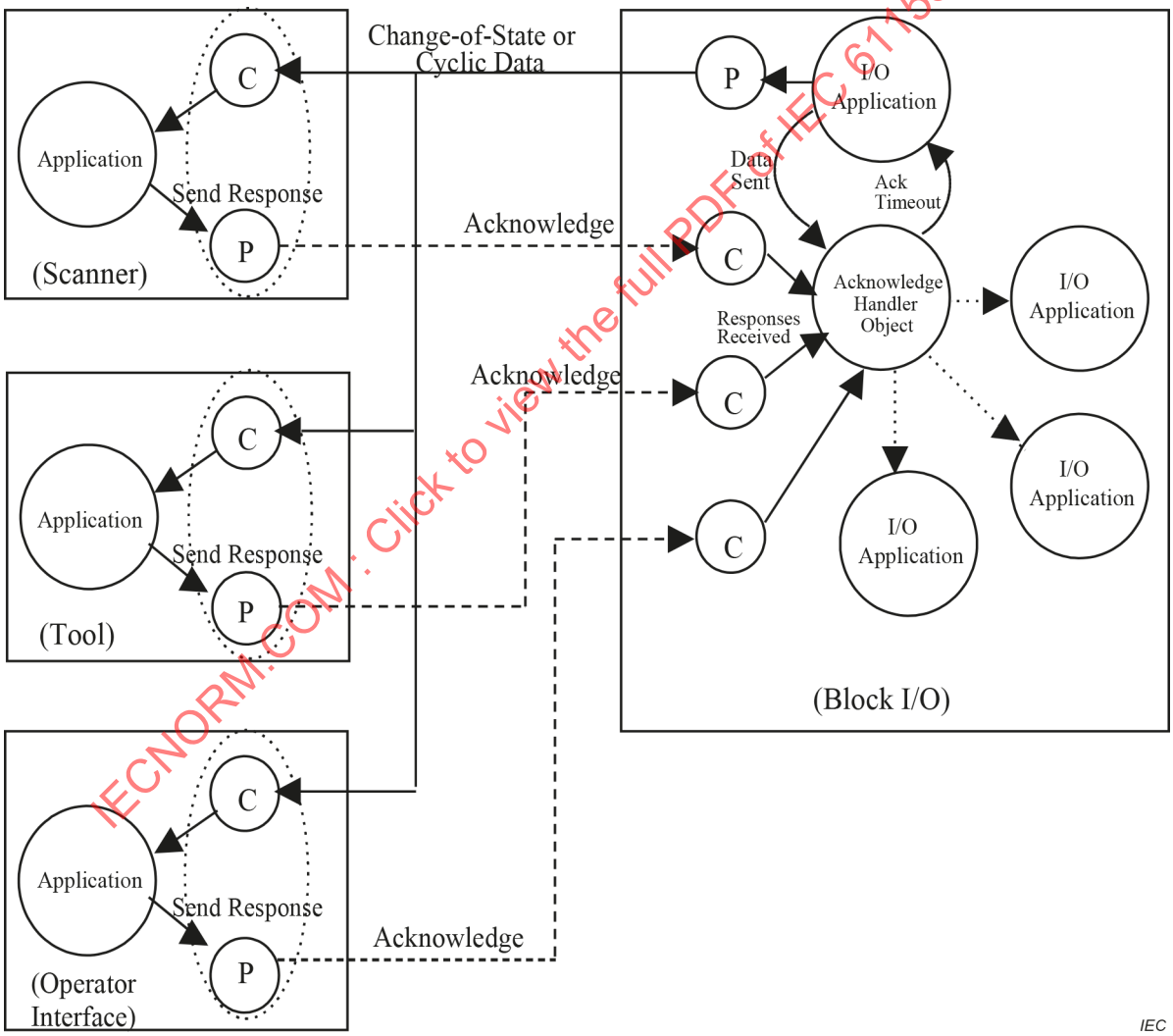

Figure 11 – Message flow in COS connection – multiple consumers

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 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.

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 = 5
2	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
200	(*)	Attribute:	Max Instance (long)
201	(*)	Attribute:	Number of Instances (long)
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:	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
30	(*)	Attribute:	DefaultLeapSeconds
31	(*)	Attribute:	AssociatedInterfaceLabels
31.1	(m)	Attribute:	NumberOfPorts
31.2.1	(m)	Attribute:	PortNumber
31.2.2	(m)	Attribute:	AssociatedInterfaceLabelPathSize
31.2.3	(m)	Attribute:	AssociatedInterfaceLabel
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
29	(*)	Ops Service:	Get_Connection_Point_Member_List

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

The Time Sync ASE (object) supports the following instance attributes. See the PTP specification in IEC 61588 for additional details. See 6.2.1.2.6.5, c) (PTP port assignments) for information concerning PTP ports and their relationship with Type 2 ports.

PTPEnable

Specifies whether the Precision Time Protocol is enabled on 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 17 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 Type 2 Time Synchronization 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 17 – 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

Specifies the current system time in units of nanoseconds. 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.

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 absolute value of the maximum amount of deviation between the local clock and the master clock in nanoseconds since last set. The value 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 device.

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 communication profile.

PortNumber

Specifies the PTP 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.

PortStateInfo

Specifies the current state 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 (log_2) seconds. See 6.2.1.2.6.5 (Type 2 PTP profile default settings and ranges) for default values.

The PortLogAnnounceInterval member of a port should be greater than or equal to its PortLogSyncInterval member. A device may reject a set attribute request that does not meet this criteria with General Status code “Invalid Attribute Value”. If no error is returned, the behavior is vendor specific.

This 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 (log_2) seconds. See 6.2.1.2.6.5 (Type 2 PTP profile default settings and ranges) for default values.

The PortLogSyncInterval member of a port should be less than or equal to its PortLogAnnounceInterval member. A device may reject a set attribute request that does not meet this criteria with General Status code “Invalid Attribute Value”. If no error is returned, the behavior is vendor specific.

This 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 (Type 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 (Type 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 (Type 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, i.e. the IEEE OUI (Organization Unique Id) for the manufacturer.

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.

EXAMPLE "ACME Manufacturing;C2901;123456"

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.

EXAMPLE "1.2;2.3;3.0.1" or "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.

EXAMPLE Sensor-1; Section-7 Cabinet-2 Rack-3

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 volatile 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, r).

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

AssociatedInterfaceObjects

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

Specifies the Type 2 port or link object interface paths for all PTP ports for a device. If a label for the interface object is required, the attribute AssociatedInterfaceLabels shall be used.

PTP port numbers shall start with 1 and be sequential, according to IEC 61588.

Therefore, this attribute is mandatory if any of the following conditions are true, otherwise it is optional.

- A device supports more than one Type 2 port and it is not possible for the PTP and Type 2 port numbers to be the same.
- More than one PTP port is associated with a single Type 2 port (for example PRP).

See 6.2.1.2.6.5, c) (PTP port assignments) for examples where the PTP port and Type 2 port numbers could be different.

If a single PTP port is associated with a single Type 2 port, the AssociatedInterfaceObjects attribute shall specify the Port object instance that represents the Type 2 port (i.e. 20 F4 24 xx, where xx is the Port object instance).

If there are two or more PTP ports associated with a single Type 2 Ethernet port (for example in the PRP case), the AssociatedInterfaceObjects attribute shall specify the Ethernet Link object instance (i.e. 20 F6 24 xx, where xx is the Ethernet Link object instance).

If there are two or more PTP ports associated with any other single Type 2 port, the AssociatedInterfaceObjects Attribute shall specify the logical path segments to the other Link object instance (i.e. 20 xx 24 yy).

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

DefaultLeapSeconds

Specifies the number of leap seconds for the "Current UTC Offset" field of the Precision Time Protocol (Default Data Set). This value is used by a master-capable device when operating as a grandmaster, unless some other means is available to obtain the UTC offset (for example GPS, user interface).

This instance attribute is mandatory for master capable devices.

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

AssociatedInterfaceLabels

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

Specifies the Type 2 port or link label paths for all PTP ports for a device. A client can get the interface labels from the returned label paths. The attribute AssociatedInterfaceObjects shall be used to return the Type 2 interface object paths.

If a single PTP port is associated with a single Type 2 port, the AssociatedInterfaceLabels attribute shall specify the Port object instance, PortName attribute, that represents the Type 2 port (i.e. 20 F4 24 xx 30 04, where xx is the Port object instance).

If there are two or more PTP ports associated with a single Type 2 Ethernet port (for example in the PRP case), the AssociatedInterfaceLabels attribute shall specify the Ethernet Link object instance, InterfaceLabel attribute (i.e. 20 F6 24 xx 30 0A, where xx is the Ethernet Link object instance).

If there are two or more PTP ports associated with any other single Type 2 port, the AssociatedInterfaceLabels Attribute shall specify the logical path segments to the other Link object instance (i.e. 20 xx 24 yy).

If there are two or more PTP ports associated with any other single Type 2 port, the AssociatedInterfaceLabels attribute shall specify the logical path segments to an attribute representing that other link object instance's interface label (i.e. 20 xx 24 yy 30 zz, where xx is the link object, yy is the link object instance, and zz is the interface label attribute. The type of this attribute shall be SHORT_STRING.

This attribute is mandatory if any of the conditions detailed for AssociatedInterfaceObjects are true, 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

Get_Attribute_Single

This service shall support the use of a connection point in the service path when Diagnostic connection point(s) are implemented.

Get_Connection_Point_Member_List

This service is used to get member paths of a Connection Point structure. It is **mandatory** if the Diagnostic connection point(s) are implemented, else it is **optional**.

6.2.1.2.6.5 Specific behavior for the Time Sync object

a) System Time definition

The starting date/time for Type 2 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 Type 2 network communication profile. 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 61158-6-2,4.1.8.6.1.2.

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 Type 2 protocols, the PTP and Type 2 port numbers should be the same where possible.

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

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

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

d) Type 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 Type 2 network communication profiles is specified in bullets e) to p). It is based on the Delay Request-Response default PTP profile defined in IEC 61588.

e) Type 2 PTP profile – Profile identification

Table 18 identifies the Type 2 PTP profile. The IEC 61588 Profile identifier shall be used on an IEC 61588 network and the IEC 62439-3 PRP identifier on a PRP network.

Table 18 – Profile identification

Profile identifier	Usage	Specification
PTP profile description		Type 2 PTP Profile
Version		1.0 or 1.1
Profile identifier	IEC 61588:2009 IEC 61588:2021 With PRP [a]	00-21-6C-00-01-00 00-21-6C-00-01-01 00-15-4E-00-01-50
Specifying organization		ODVA: Web: www.odva.org

[a] Profile identifier when operating over a PRP network (see attribute 24).

f) Type 2 PTP profile – Default settings and ranges

The Type 2 PTP profile shall support the PTP default and range settings as defined in Table 19. 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 of acceptable values for IEC 61588.

Table 19 – 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) Type 2 PTP profile – Profile transports

The supported IEC 61588 PTP transport is specified in the network communication profile.

h) Type 2 PTP profile – PTP domain

A Type 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 Type 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) Type 2 PTP profile – PTP node management

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

j) Type 2 PTP profile – Transparent Clock

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

k) Type 2 PTP profile – Path delay mechanism

Each device implementing Type 2 Time Synchronization shall implement the delay request-response path delay measurement mechanism.

l) Type 2 PTP profile – Recommended PTP clock settings

Table 20 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 20 – 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	Oscillator	Oscillator	Oscillator
Priority1	128	128 (not settable)	128	128
Priority2	128	128 (not settable)	128	128

m) Type 2 PTP profile – TTL – Time to Live

TTL is only relevant for Type 2 network communication profiles based on Ethernet. By default, the TTL value in the UDP/IPV4 packet shall be set to 1. 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) Type 2 PTP profile – Hand_Set clock quality management

Some Type 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 could for example powerup with an unknown time. Once set the time is known and to within a specified tolerance of UTC time.

Table 21 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 21 – 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) Type 2 PTP profile – Support for Device Level Ring and path reconfiguration

Devices that support ring topologies and require high synchronization accuracy, such as motion devices, 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 Type 2 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 Type 2 Time Synchronization support.

In DLR, a topology change requires that Type 2 Time Synchronization 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 Type 2 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 12. 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.

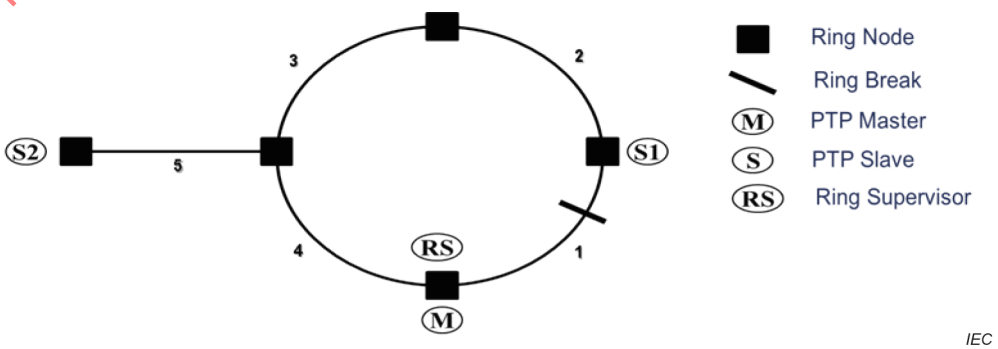

Figure 12 – Path Reconfiguration in a ring topology

p) Type 2 PTP profile – Path Reconfiguration Signalling message

The Path Reconfiguration Signalling message is issued by a device to notify that a network reconfiguration has been detected. This message is a Type 2 PTP profile specific Signalling message. DLR gateway devices will issue this message when a network reconfiguration has been detected. DLR end nodes will receive this message and recalculate the mean path delay using the delay request-response mechanism. Signaling messages in general are defined in IEC 61588, and the specific Path Reconfiguration Signaling Message is specified in Table 22. See the previous item o) for application of this message.

Table 22 – Path Reconfiguration Signalling message

Signalling TLV	Field description	Field value
TLV Type	Organization extension	03
Organization ID	Type 2 Time Synchronization ID	00 21 6C
TLV Subtype	Path reconfiguration	01

q) Type 2 Time Synchronization over Parallel Redundancy Protocol (PRP)

The IEC 61588 Precision Time Protocol (PTP) over a PRP network is specified in IEC 62439-3:2016, Annex A and Annex C.

All devices that support PRP and Type 2 Time Synchronization shall implement the Layer 3 End-to-End (IEC 62439-3:2016, Annex C, L3E2E) doubly attached clock model (paired port) for PRP.

Figure 13 shows an example of a PRP network.

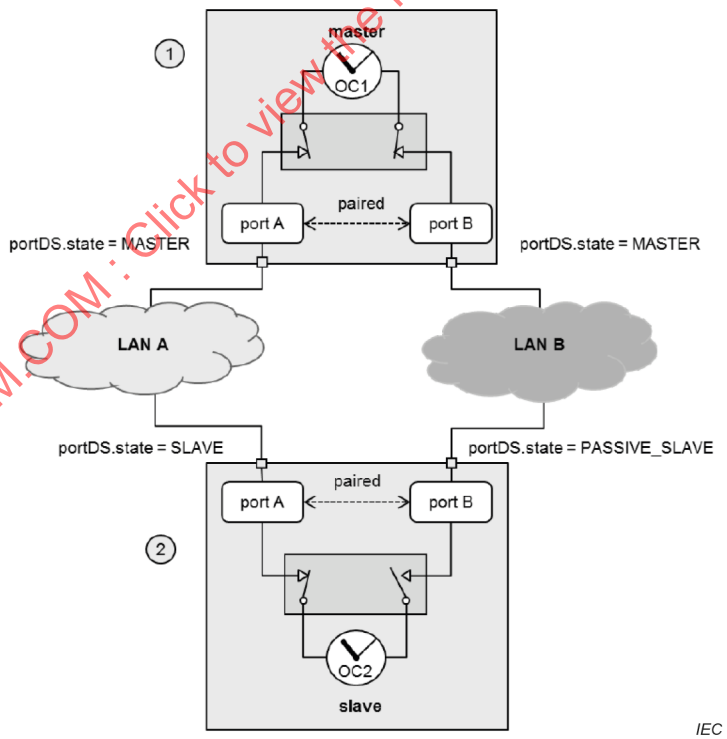

Figure 13 – Doubly attached clocks in a PRP network

Both paired ports of a doubly attached clock function as either SLAVE or MASTER when properly connected and operating on LAN A and LAN B redundant networks.

For master operation, both ports, port A and B, operate as MASTER ports as defined by the IEC 61588 PTP protocol.

For slave operation, one port is the active port and operates as defined by the IEC 61588 PTP protocol. The active port tunes the clock and reports its state as SLAVE. The other port is passive and reports its state as PASSIVE_SLAVE. The passive port also measures path delay and maintains close synchronization to the active port, so that a network failure on the active port will result in a smooth clock transition from passive to active slave.

The PASSIVE_SLAVE state is reported as value 10 of attribute 12 (PortState) of the Time Sync object.

For backward compatibility with IEC 61588:2009 management tools, the PASSIVE_SLAVE state is reported as value 7 (PASSIVE), when reporting the port state of the port data set (see IEC 62439-3).

r) Clock model

Type 2 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 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 can 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 can 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 14 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.

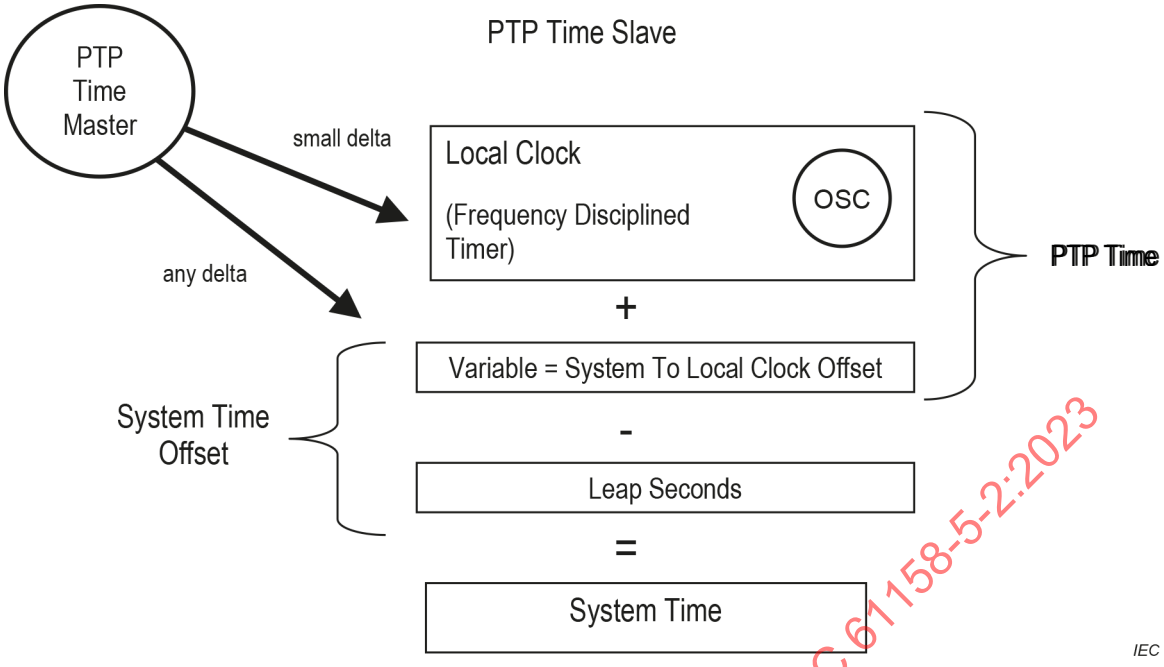

Figure 14 – Type 2 Time Synchronization offset clock model

Figure 15 shows a system of devices each implementing the Type 2 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.

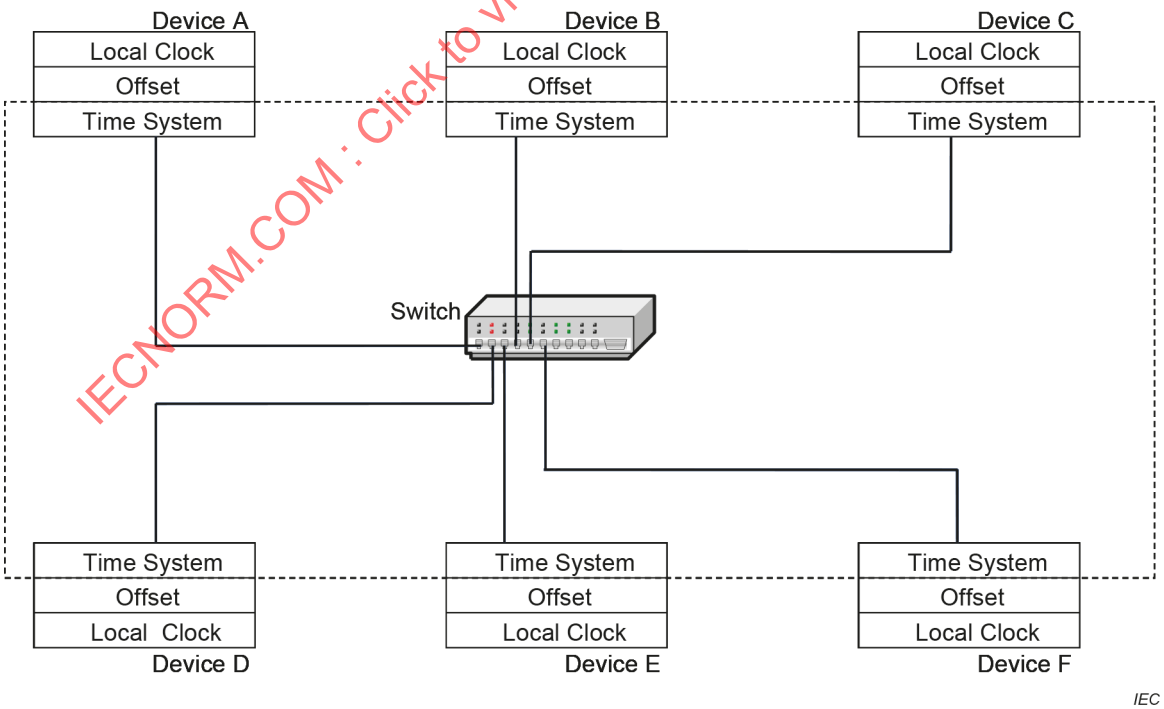

Figure 15 – Type 2 Time Synchronization system with offset clock model

s) 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 can be required if a step has occurred in System Time.

Type 2 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 can be inconsistent with a timestamp on another node. Or a timestamp taken at one instance in time can 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 step compensation algorithms are described in t), u) and v).

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

t) Step compensation algorithms (general)

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.

u) 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.

v) 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.

w) Group startup sequence

Type 2 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 16.

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.

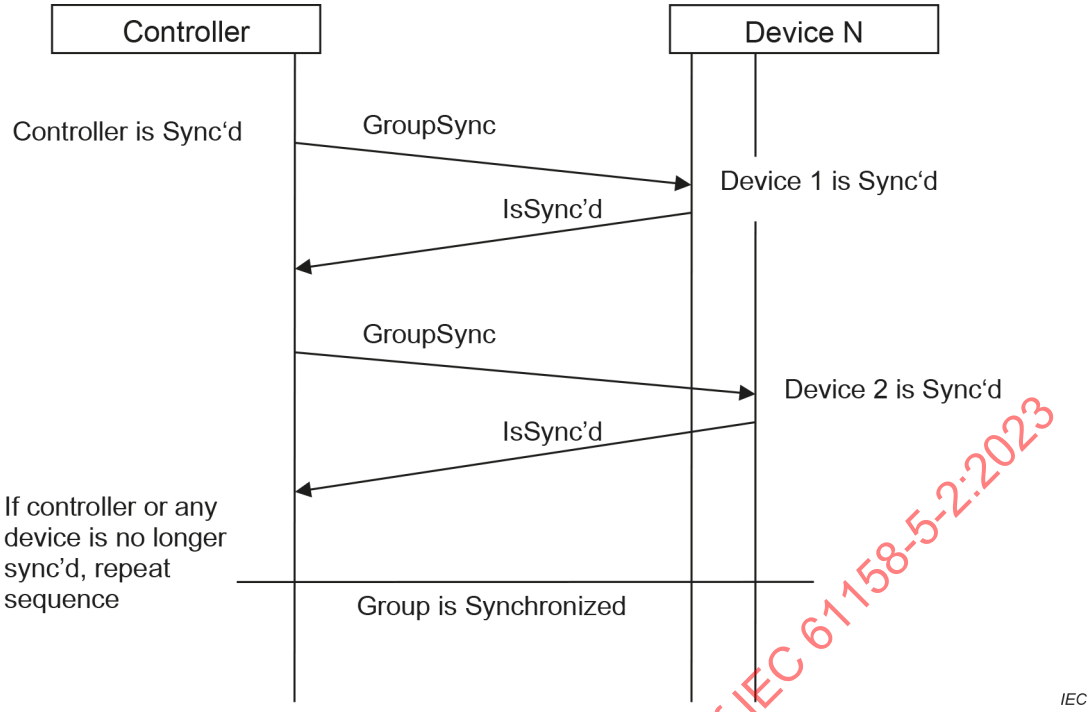

Figure 16 – Type 2 time synchronization group startup sequence

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	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
8	(m)	Attribute:	Parameter Class Descriptor
9	(m)	Attribute:	Configuration Assembly Instance
10	(*)	Attribute:	Native Language
200	(*)	Attribute:	Max Instance (long)
201	(*)	Attribute:	Number of Instances (long)

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)

Max Instance (short/long)

This attribute is mandatory. The device shall implement the long version if it supports instances greater than 65 535, else it shall implement the short version.

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, otherwise it is optional.

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

Type 2 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 23 is the state event matrix for the Parameter object. The behavior of the Parameter object is illustrated in Figure 17.

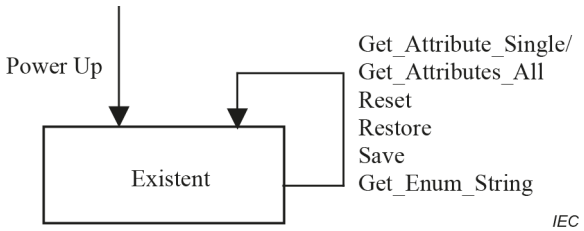

Figure 17 – Parameter object state transition diagram

Table 23 – 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
Get_Connection_Point_Member_List

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)

Send_Receive_Fragment (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, with the exception of "communication related problem". 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 heartbeat messages.

The following guidelines apply to Status code selection unless an object definition provides more specific direction:

- the most specific Status code applicable to a given error condition shall be returned;
- no order of error code evaluation is required; when more than one error condition applies to a request, devices may report any of the Status codes that apply.

Clients shall tolerate any Status code returned by the server, including any variable-sized Extended status.

6.2.1.3.3 FAL service status codes

Table 24 specifies the range of possible status codes.

Table 24 – Status codes

Name	Description and meaning of status code
Success	Service was successfully performed by the object specified
Communication related problem	A communications subsystem related problem was detected along the communications path
Resource unavailable	Resources needed for the object to perform the requested service were unavailable
Invalid parameter value	See status code “invalid parameter”, 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

Name	Description and meaning of status code
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 instance does not exist	The object instance specified does not exist in the device. This error code shall be used instead of "Path Destination Unknown" when the class is supported but the specific instance does not exist.
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
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 response 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.

Name	Description and meaning of status code
Group 2 only server general failure	Reserved for specific network communication profiles
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 when a more specific Status code applies e.g. "Attribute not settable" or "Member not settable".
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 25.

Table 25 – 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(=)

Parameter name	Req	Ind	Rsp	Cnf
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 26.

Table 26 – 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 Extended 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" 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" 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" 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" 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 27.

Table 27 – 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 Response Structures			M	M(=)
Attribute ID			M	M(=)
Attribute 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.

Attribute count

This parameter indicates the number of attribute IDs in the Attribute List parameter below.

Attribute list

This parameter contains the list of the attribute IDs for the attributes that are being requested from the specified class or object instance.

Result(+)

This selection type parameter indicates that the service request succeeded, at least partially.

Service status

This parameter indicates success, "Partial transfer" or "Attribute list error" (see service procedure in 6.2.1.3.6.3).

Attribute count

This parameter indicates the number of attributes actually returned in the Attribute Response Structures 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 response structures

This parameter returns a stream of information containing all or part of the requested attributes. Attribute data returned shall be retrieved and packed in the sequence specified in the request. Each Attribute Response Structure included in the response shall be complete.

Attribute ID

This parameter contains the attribute ID corresponding to the data being returned in Attribute Data.

Attribute status

This parameter indicates the individual status information for the requested attribute (see Table 24).

Attribute data

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). When an Attribute Response Structure has an Attribute Status value other than Success, then the Attribute Data shall be omitted for that structure.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 24).

6.2.1.3.6.3 Service procedure

The requested attributes shall be specified using a list of attribute IDs.

If any of the Attribute Response Structures have an Attribute Status value other than Success, then General Status code "Attribute list error" shall be returned for this service.

The number of attributes returned shall be reported within the response. The server may return fewer than the requested number of attributes if there is not enough space for all Attribute Response Structures. If a partial response is returned and all Attribute Status values are Success, then General Status code "Partial transfer" shall be returned. However, if one or more Attribute Status values are not Success, General Status code "Attribute list error" shall be returned instead.

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 28.

Table 28 – 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 Request Structures	M	M(=)		
Attribute ID	M	M(=)		
Attribute Data	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Count			M	M(=)
Attribute Response Structures			M	M(=)
Attribute ID			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 Request Structures parameter below.

Attribute Request Structures

This parameter contains a stream of information containing the actual list of attributes to be updated and the corresponding update data.

Attribute ID

This parameter specifies the attribute ID corresponding to the attribute to be updated with the contents of Attribute Data. The client is unable to specify sub-elements of an attribute.

Attribute data

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, at least partially.

Service status

This parameter indicates success, "Partial transfer" or "Attribute list error" (see service procedure in 6.2.1.3.7.3).

Attribute count

This parameter indicates the number of attribute status returned in the Attribute Status List parameter below.

Attribute Response Structures

This parameter returns a stream of information containing status information for each attribute listed in the service request. The Attribute Response Structures shall be in the order specified in the request for each attribute that was set.

Attribute ID

This parameter contains the attribute ID corresponding to the status being returned in Attribute Status.

Attribute status

This parameter indicates the individual status information for the specified attribute (see Table 24).

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 24).

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).

Each set activity shall be performed in the order specified in the list.

A server shall not attempt to set an attribute for which it cannot return a complete Attribute Response Structure. The server shall stop processing additional Set requests and return General Status code "Partial transfer".

When there is an error setting an individual attribute, the object may respond with one of two behaviors.

- Set attributes until error:
  - stop further processing of attributes in the request;
  - any attributes that were set shall result in Success Attribute Response Structures responses;
  - return an appropriate error for the attribute in error in the Attribute Response Structure;
  - return General Status code "Attribute list error".
- Set no attributes:
  - stop further processing of attributes in the request;
  - discard Sets prior to the first error;
  - return Attribute Response Structure with an appropriate error only for the attribute in error;
  - return General Status code "Attribute list error".

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 29.

Table 29 – 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 30.

Table 30 – 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 31.

Table 31 – 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(=)

Parameter name	Req	Ind	Rsp	Cnf
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 32.

Table 32 – 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 33.

Table 33 – 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, connection point 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). Subattribute addressing is not supported for connection points.

6.2.1.3.13.2 Service primitives

The service parameters for this service are shown in Table 34.

Table 34 – 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 Type 2 network communication 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 instance 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 35.

Table 35 – 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 Type 2 network communication 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 instance 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.

NOTE Even though the Instance ID value to use as the lower bound of the search is provided in the application path, the service is not supported at the instance level.

6.2.1.3.15.2 Service primitives

The service parameters for this service are shown in Table 36.

Table 36 – 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 18 provides a general example of this service.

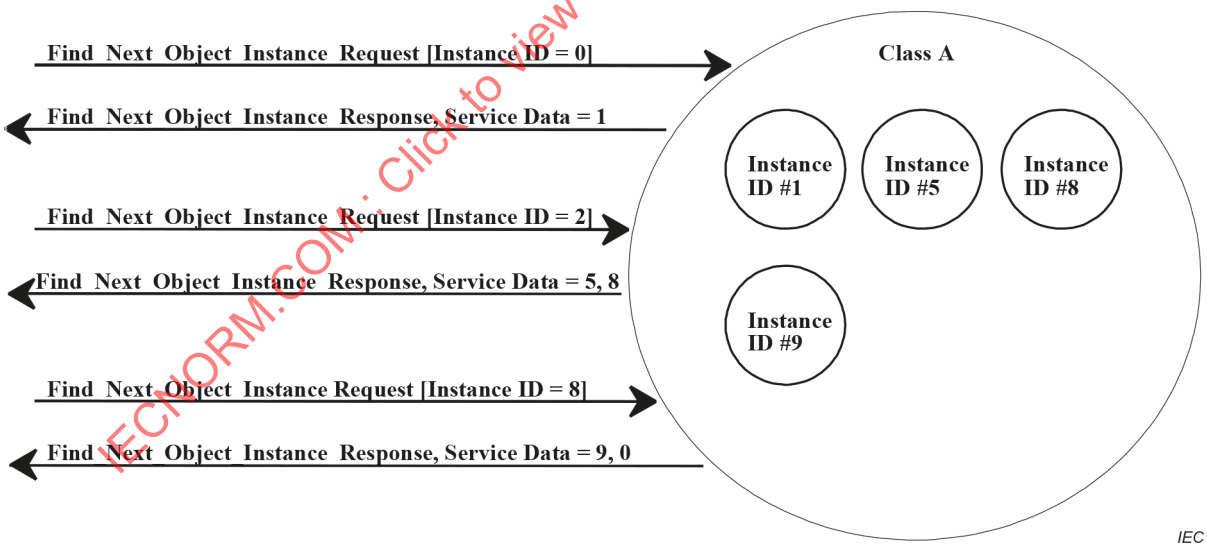

Figure 18 – Example of Find_Next_Object_Instance service

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 37.

Table 37 – 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 38.

Table 38 – 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 39.

Table 39 – 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 40.

Table 40 – 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 41.

Table 41 – 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 24).

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 there is no member, 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.
- 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 extended 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 42.

Table 42 – 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 24).

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 43.

Table 43 – 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 24).

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 44.

Table 44 – 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 24).

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 there is no member, an error response with status "Invalid Member ID" is returned.
- 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 45.

Table 45 – 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 24).

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 46.

Table 46 – 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 24).

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 47.

Table 47 – 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 24).

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 48.

Table 48 – 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 24).

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 49.

Table 49 – 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 24).

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 50.

Table 50 – 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 24).

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 Type 2 network. 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 Type 2 network, 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 51.

Table 51 – 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 24).

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 can cause an error response such as "Response Data Too Large" to be returned.

  Four octets shall be reserved by the Message Router in the response packet for each embedded service to satisfy this requirement (Response service code octet, pad octet, General Status octet and Extended Status size octet).
- 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 "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.

6.2.1.3.31 Get_Connection_Point_Member_List

6.2.1.3.31.1 Service overview

The Get_Connection_Point_Member_List service returns an array of EPATHs and associated size (in bits) of the value for the member attributes that define the structure (including pad bits) of the connection point specified in the service path.

NOTE The format is essentially the same as the combined instance attributes 1 and 2 of the Assembly ASE (object).

6.2.1.3.31.2 Service primitives

The service parameters for this service are shown in Table 52.

Table 52 – Get_Connection_Point_Member_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		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Item Count			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. 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.

Item Count

This parameter indicates the number of STRUCTs returned in the Members parameter below.

Members

This parameter contains an array of service response structures which contain EPATHs and associated size (in bits) of the value for the member attributes that define the structure (including pad bits) of the connection point specified in the service path.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 24).

6.2.1.3.31.3 Service procedure

The following list details requirements associated with the Get_Connection_Point_Member_List service.

- The number of EPATHs returned shall be reported as the first value within the response.
- The EPATHs returned shall be placed in the sequence they are specified in the connection point definition.
- Padding represented in the connection point definition shall be represented as separate members with a null (0 length) EPATH.
- The EPATHs returned for data elements shall be the Class/Instance/Attribute of each member of the connection point structure. If a device uses full Parameter object instances to enumerate the value of the member(s), an implementation may substitute the EPATHs to attribute 1 of the Parameter object instance for the member(s).

6.2.1.3.32 Send_Receive_Fragment

6.2.1.3.32.1 Service overview

The Send_Receive_Fragment service provides a mechanism to transfer fragments of a service request and/or response that would otherwise be too large to deliver in a single Type 2 packet. A simple fragmentation protocol is used to accumulate multiple fragments before passing the service on to the destination object for further processing and splitting the response into multiple fragments and sending them back to the requestor. The original service becomes an embedded service within the Send_Receive_Fragment service.

The Send_Receive_Fragment service can be used to fragment the request, the response, or both the request and the response in the same session.

The Send_Receive_Fragment service session consists of two phases:

- in phase 1 the client sends the fragmented request;
- in phase 2 the target returns the fragmented response.

6.2.1.3.32.2 Service primitives

The service parameters for this service are shown in Table 49.

Table 53 – Send_Receive_Fragment 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		
Fragmentation Version	M	M(=)		
Request Fragmentation Flags	M	M(=)		
Request Info	M	M(=)		
Embedded Service Request Fragment	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Fragmentation Version			C	C(=)
Response Fragmentation Flags			C	C(=)
Response Info			C	C(=)
Embedded Service Response Fragment			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.

Fragmentation Version

This parameter contains the version of the fragmentation protocol. Currently only version 1 is defined.

Request Fragmentation Flags

This parameter contains contains flags controlling the request fragmentation packet.

In phase 2, for requests to get the next fragment response, this parameter shall always specify Middle fragment.

Request Info

For the first fragmentation request of phase 1, this parameter contains the total number of octets in the embedded service request that will be transferred to the target (total request size).

For the subsequent fragmentation requests of phase 1, this parameter contains the zero-based offset of the first octet of the Embedded Service Request Fragment within the embedded service request.

For requests of phase 2 to get the next fragment response, this parameter shall be set to zero.

Embedded Service Request Fragment

This conditional parameter is mandatory for phase 1, and not present for phase 2.

For the client's fragmented requests of phase 1, this parameter holds a single fragment of the embedded service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

The following conditional parameters are mandatory for phase 2, and not present for phase 1.

Fragmentation Version

This parameter contains the version of the fragmentation protocol. Currently only version 1 is defined.

Response Fragmentation Flags

This parameter contains contains flags controlling the response fragmentation.

Response Info

For the initial fragmentation response, this parameter contains the total number of octets in the embedded service response that will be transferred to the client.

For subsequent fragmentation responses, this parameter contains the zero-based offset of the first octet of the Embedded Services Response Fragment within the embedded service response.

Embedded Service Response Fragment

This parameter holds a single fragment of the embedded target response.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 24).

6.2.1.3.32.3 Service procedure

a) General

The Send_Receive_Fragment service can only be sent using an explicit messaging connection. If the service is received via an unconnected message a response shall be returned with General Status Code "Communication related problem", Extended Status Code "This service requires a connection".

The established connection size should be used for all packets that contain an Embedded Service Request Fragment or an Embedded Service Response Fragment except when the packet is a last fragment.

From a modeling perspective, the target Message Router accumulates the request fragments and when all have been received, sends the reassembled embedded service request to the destination object (phase 1) and receives the response from the destination object. The target Message Router fragments the response to send back to the client. The client accumulates the response fragments and when all have been received, processes the reassembled embedded response (phase 2).

A fragmentation session starts when receiving a request with the First bit set and terminates when aborted, when a request is received with the First bit set, the connection times out or is closed, or a response with the Last bit set is returned. Termination includes freeing of session resources.

b) Transfer a Fragmented Request – Phase 1

When the first fragment is received, the Request Info parameter contains the total message request size. The target can use the size parameter to allocate a receive buffer large enough to hold the entire request and to verify all the fragments have been received. The first fragment is used to deliver the first portion of the embedded service request.

The middle fragments and last fragments are used to deliver the rest of the embedded service request in order. The Request Info parameter contains the offset within the receive buffer of where to place the received fragment. The contents of the Request Info parameter are used to verify that all fragments are received in the correct order.

When the last fragment is received the target Message Router shall pass the reassembled embedded service request to the target object for further processing and then transition to phase 2.

c) Transfer a Fragmented Response – Phase 2

The target Message Router is responsible for fragmenting the response being returned to the client.

When the first fragment response is sent, the Response Info parameter contains the total response size. The client can use the size parameter to allocate a receive buffer large enough to hold the entire response and to verify all the fragments have been received. If the client is unable to provide a buffer large enough for the response the client should terminate the fragmentation session by setting the Abort flag.

The middle fragments and last fragments are used to deliver the rest of the embedded service response in order. The Response Info parameter contains the offset within the buffer of where to place the received data. The contents of the Response Info parameter are used for verifying that all the fragments are received in the correct order.

d) Send_Receive_Fragment Service Exception Handling – Prior to Phase 2 (Request)

See IEC 61158-6-2, 4.1.8.3.3.2 for the list of object-specific General and Extended Status Codes.

If processing a first fragment request and the Request Info parameter value (total request size) is too large the target shall return a General Status Code “Resource unavailable”, Extended Status Code “Request Info Too Large”.

If processing a first fragment request and the Request Info parameter value (total request size) is 0 the target shall return a General Status Code “Service fragmentation…”, Extended Status Code “Fragment Received With Invalid Request Info”.

If a first fragment request is received and the target does not have resources for the number of simultaneous fragmentation sessions it shall return a General Status Code “Resource unavailable”, Extended Status Code “Out Of Fragmentation Sessions”.

If an incorrect offset (Request Info) is received then the target shall respond with a General Status code “Service fragmentation…”, Extended Status Code “Request Info is Out of Sequence”.

If a Middle or Last fragment is received when not having an active fragmentation session the target shall respond with a General Status code “Service fragmentation…”, Extended Status Code “No Active Fragmentation Session”.

If a Middle fragment is received when expecting a Last the target shall respond with a General Status code “Service fragmentation…”, Extended Status Code “Middle Fragment Received When Expecting a Last Fragment”.

If a Last fragment is received when expecting a Middle the target shall respond with a General Status code “Service fragmentation…”, Extended Status Code “Last Fragment Specified in Request When Expecting a Middle Fragment – Phase 1”.

If a fragment is received and the offset plus Embedded Service Request Fragment size is greater than the total request size the target shall respond with a General Status code “Service fragmentation…”, Extended Status Code “Invalid Size – Too Much Data”.

If a request is received with both First and Last bits set and the Request Info (total request size) does not equal the Embedded Service Request Fragment size, the target shall return General Status code “Service fragmentation…”, Extended Status code “Invalid Size – Non-Fragmented Size Mismatch”.

If the Fragmentation Version parameter is not supported the target shall respond with a General Status code “Invalid Parameter”, Extended Status Code “Unsupported Version”.

If the target receives a First fragment request during phase 1 the target shall terminate the ongoing fragmentation session and attempt to start a new fragmentation session.

A loss of the connection results in a termination of the session.

e) Send_Receive_Fragment Service Exception Handling – Phase 2 (Response)

See IEC 61158-6-2, 4.1.8.3.2 for the list of Object-specific General and Extended Status Codes.

After the Last request fragment is received, if the total assembled response is too large for the device to handle then the target shall return a General Status Code “Resource unavailable”, Extended Status Code “Assembled Response Too Large”.

When the client receives a response with the First bit set and the Response Info is too large the client should terminate the fragmentation session by sending a Send_Receive_Fragment service request with the Abort flag set.

If a non-zero value in the Request Info parameter is received the target shall respond with a General Status code “Service fragmentation…”, Extended Status Code “Fragment Received With Invalid Request Info”.

If the target receives a Last fragment request during phase 2 the target shall respond with a General Status code “Service fragmentation…”, Extended Status Code “Last Fragment Specified in Request When Expecting Middle Fragment – Phase 2”.

If the target receives a First fragment request during phase 2 the target shall terminate the ongoing fragmentation session and attempt to start a new fragmentation session.

If a request is received with both First and Last bits set and the Request Info (total request size) does not equal the Embedded Service Request Fragment size, the target shall return General Status code “Service fragmentation…”, Extended Status code “Invalid Size – Non-Fragmented Size Mismatch”.

A loss of the connection results in a termination of the session.

f) Fragmentation Session Management

The processing of fragmentation sessions is accomplished by the Fragmentation Session Manager and a state machine within each fragmentation session.

The Fragmentation Session Manager receives messages and connection state information from the Message Router and either routes each message to the appropriate session or responds directly. The Message Router delivers the connection instance with each notification to the Fragmentation Session Manager which provides a linkage between the Message Router and each session. Table 54 describes the behavior of the Fragmentation Session Manager.

Table 54 – Fragmentation Session Manager Event/Activity Matrix

Event	Connection instance linked to session	Activity
Message received with Abort (Abort = 1)	N	Return General Status code “Service fragmentation…”, Extended Status “No Active Fragmentation Session”
	Y	Send Terminate event to session
First message received with invalid Request Info (Abort = 0, First = 1, Request Info = 0)	N	Return General Status code “Service fragmentation…”, Extended Status Code “Fragment Received With Invalid Request Info”
	Y	Return General Status code “Service fragmentation…”, Extended Status Code “Fragment Received With Invalid Request Info” and send Terminate event to session
First message received with valid Request Info (Abort = 0, First = 1, Request Info != 0)	N	If all fragmentation sessions are in use return General Status Code “Resource unavailable”, Extended Status Code “Out Of Fragmentation Sessions” Else allocate a session, link connection instance, and send either Receive first of multiple fragments with valid Request Info or Receive first and last fragment with valid Request Info event depending on the state of the Last flag
	Y	Send either Receive first of multiple fragments with valid Request Info or Receive first and last fragment with valid Request Info event depending on the state of the Last flag
Middle message received (Abort = 0, First = 0, Last = 0)	N	Return General Status code “Service fragmentation…”, Extended Status “No Active Fragmentation Session”
	Y	Send either Receive Middle fragment with Request Info equal to zero or Receive Middle fragment with Request Info not equal to zero event depending on value of Request Info equal to zero or not
Last message received (Abort = 0, First = 0, Last = 1)	N	Return General Status code “Service fragmentation…”, Extended Status “No Active Fragmentation Session”
	Y	Send either Receive Last fragment with Request Info equal to zero or Receive Last fragment with Request Info not equal to zero event depending on value of Request Info equal to zero or not
Session end [a]	N	N/A
	Y	Unlink connection instance and session, set session to not in use
Message received from UCMM	N/A	Return General Status Code “Communication related problem”, Extended Status Code “This service requires a connection”
Connection Close/ Connection timeout	N	Ignore
	Y	Send Terminate event to session

[a] This event is sent from a session

Each fragmentation session shall implement its own state machine as specified in the State Transition Diagram (see Figure 19) and State Event Matrix (see Table 55).

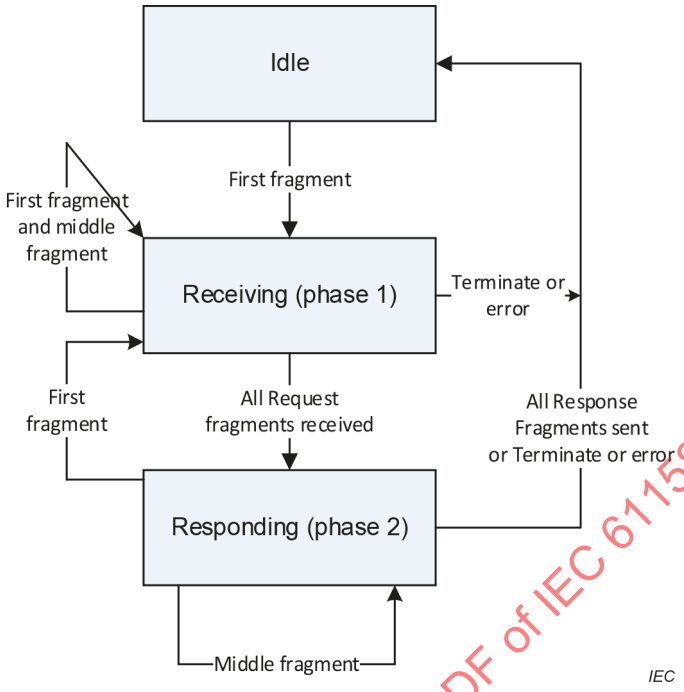

Figure 19 – State Transition Diagram for Fragmentation Session

Table 55 – Fragmentation State Event Matrix

	Fragmentation State		
Event	**Idle**	**Receiving (phase 1)**	**Responding (phase 2)**
Terminate	N/A	Terminate current fragmentation session and transition to Idle [a]	
Receive first of multiple fragments with valid Request Info (First = 1, Last = 0, Request Info not equal to 0)	Evaluate Request Info (total request size). If too large for the target to handle return General Status Code “Resource unavailable”, Extended Status Code “Request Info Too Large” and stay in Idle [a] Else Initialize session resources, accept fragment data, and transition to Receiving (phase 1)	Terminate current fragmentation session Evaluate Request Info (total request size). If size is too large for the target to handle, return General Status Code “Resource unavailable”, Extended Status Code “Request Info Too Large” and transition to Idle [a] Else Initialize session resources, accept fragment data, and stay in (or transition to) Receiving (phase 1)	

	Fragmentation State		
Event	**Idle**	**Receiving (phase 1)**	**Responding (phase 2)**
Receive first and last fragment with valid Request Info (First = 1, Last = 1, Request Info not equal to 0)	If Request Info (total request size) does not equal Embedded Service Request Fragment size, return General Status code “Service fragmentation…”, Extended Status code “Invalid Size – Non-Fragmented Size Mismatch” and stay in Idle [a] Else Initialize session resources, accept fragment data, forward the embedded request to the target object and transition to Responding (phase 2)	Terminate current fragmentation session If Request Info (total request size) does not equal Embedded Service Request Fragment size, return General Status code “Service fragmentation…”, Extended Status code “Invalid Size – Non-Fragmented Size Mismatch” and transition to Idle [a] Else Initialize session resources, accept fragment data, forward the embedded request to the target object, and stay in (or transition to) Responding (phase 2)	
Receive Middle fragment with Request Info not equal to zero (First = 0, Last = 0, Request Info not equal to 0)	N/A	If the expected buffer offset is incorrect, terminate current fragmentation session, return General Status code “Service fragmentation…”, Extended Status Code “Request Info is Out of Sequence”, and transition to Idle [a] Else If offset plus Embedded Service Request Fragment size is greater than or equal to the total request size terminate ongoing fragmentation session, return General Status code “Service fragmentation…”, Extended Status code “Middle Fragment Received When Expecting a last Fragment”, and transition to Idle [a] Else Append received data to buffer	Terminate current fragmentation session, return General Status code “Service fragmentation…”, Extended Status Code “Fragment Received With Invalid Request Info”, and transition to Idle [a]
Receive Middle fragment with Request Info equal to zero (First = 0, Last = 0, Request Info equal to 0)	N/A	Terminate current fragmentation session, return General Status code “Service fragmentation…”, Extended Status Code “Fragment Received With Invalid Request Info”, and transition to Idle [a]	If the remaining response fits in this fragment set Response Fragmentation Flags to Last, return remaining Response Data, and transition to Idle [a] Else Set Response Fragmentation Flags to Middle, return next fragment data, and stay in Responding (phase 2)

	Fragmentation State		
Event	**Idle**	**Receiving (phase 1)**	**Responding (phase 2)**
Receive Last fragment with Request Info equal to zero (First = 0, Last = 1, Request Info equal to 0)	N/A	Terminate current fragmentation session, return General Status code “Service fragmentation…”, Extended Status Code “Fragment Received With Invalid Request Info”, and transition to Idle [a]	Terminate current fragmentation session and return General Status code “Service fragmentation…”, Extended Status Code “Last Fragment Specified in Request When Expecting Middle Fragment – Phase 2” and transition to Idle [a]
Receive Last fragment with Request Info not equal to zero (First = 0, Last = 1, Request Info not equal to 0)	N/A	If the expected buffer offset is incorrect, terminate current fragmentation session, return General Status code “Service fragmentation…”, Extended Status Code “Request Info is Out of Sequence” and transition to Idle [a] Else If offset plus Embedded Service Request Fragment size is less than to the total request size terminate ongoing fragmentation session, return General Status code “Service fragmentation…”, Extended Status code “Last Fragment Specified in Request When Expecting a Middle Fragment” and transition to Idle [a] Else If offset plus Embedded Service Request Fragment size is greater than the total request size terminate current fragmentation session, return General Status code “Service fragmentation…”, Extended Status code “Invalid Size – Too Much Data”, and transition to Idle [a] Else Forward the reassembled embedded request to the target object and transition to Responding (phase 2)	Terminate ongoing fragmentation session and return General Status code “Service fragmentation…”, Extended Status Code “Last Fragment Specified in Request When Expecting Middle Fragment – Phase 2” and transition to Idle [a]

	Fragmentation State		
Event	**Idle**	**Receiving (phase 1)**	**Responding (phase 2)**
Target object response	N/A	N/A	If the response from the object is too large return General Status code “Resource unavailable”, Extended Status Code “Assembled Response Too Large”, and transition to Idle [a] Else If the target object returned an error return General Status code “Embedded Service Error”, include the target object error in the Response Data, and transition to Idle [a] Else If the response fits in one fragment set Response Fragmentation Flags to First and Last, return all Response Data, and transition to Idle [a] Else Set Response Fragmentation Flags to First, return first fragment data, and stay in Responding (phase 2)
[a] Release all session resources and send Session End event to Fragmentation Session Manager.			

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.

(*) 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	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
200	(*)	Attribute:	Max Instance (long)

201	(*)	Attribute:	Number of Instances (long)

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
29	(*)	Ops Service:	Get_Connection_Point_Member_List

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 could 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 is not necessarily a good indication of what is 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**.

This service is used to read a Connection Manager attribute or a connection point. It shall support the use of a connection point in the service path when Diagnostic connection point(s) are implemented.

Get_Connection_Point_Member_List

This service is **mandatory** if the Diagnostic connection point(s) are implemented, 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, or VERIFIED.

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 behavior 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 56.

Table 56 – 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.

The Forward_Open and Large_Forward_Open services shall be sent using the UCMM or a local explicit messaging connection only. They shall not be sent over a bridged explicit messaging connection. On subnets which support UCMM messaging, the recipient (either a target or a router) shall support these services using the UCMM and, in addition, may support these services over a local explicit messaging connection. On subnets which do not support UCMM messaging, these services shall be sent using a local explicit messaging 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 57.

Table 57 – 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 Forward_Close service shall be sent using the UCMM or a local explicit messaging connection only. It shall not be sent over a bridged explicit messaging connection. On subnets which support UCMM messaging, the recipient (either a target or a router) shall support this service using the UCMM and, in addition, may support this service over a local explicit messaging connection. On subnets which do not support UCMM messaging, this service shall be sent using a local explicit messaging connection.

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 “Communication related problem” and Extended Status "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 “Communication related problem” and an Extended Status code “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 can 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 a router 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 can have timed out at this router). An improperly formatted Connection_Path shall, and a Connection_Path mismatch, can cause the service to be unsuccessful and return an error (see connection path error conditions below).

If either a target or router 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 “Too much data” 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 “Not enough data” 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 “Communication related problem” and an Extended Status code of either “Error In Forward Close Service Connection Path Mismatch” or “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 router 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 58.

Table 58 – 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 router 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 routers. When a router 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 router from the UCMM response generated by the target node or by an router 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 router to router 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 59.

Table 59 – 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 60.

Table 60 – 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(=)		

Parameter name	Req	Ind	Rsp	Cnf
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 61.

Table 61 – 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 the ownership status.

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.

Unless otherwise noted, all services and attributes of the Connection object are accessible using Explicit Messaging.

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	(*)	Attribute:	Max Instance (short)
3	(*)	Attribute:	Number of Instances (short)
4	(o)	Attribute:	Optional attribute list
5	(o)	Attribute:	Optional service list
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
8	(*)	Attribute	Connection Request Error Count