IEC 61158-5-2

Edition 2.0 2010-08

INTERNATIONAL STANDARD

colour inside

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

IEC 61158-5-2:2010(E)

IEC 61158-5-2

Edition 2.0 2010-08

INTERNATIONAL STANDARD

colour inside

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

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE XH

ICS 25.04.40; 35.100.70; 35.110

ISBN 978-2-88912-105-2

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.

NOTE 1 Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in Type combinations as specified explicitly in the IEC 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

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

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

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

- Clause 2 and Bibliography: update of normative and bibliographic references

- subclause 6.1.3: update lists with new objects
- subclause 6.2.1.2.6: update of the Time Sync ASE/object (to match new IEC 61558)
- new subclause 6.2.1.2.7: new Parameter ASE/object
- subclause 6.2.1.3: update/add services for Time Sync and Parameter ASEs
- subclause 6.2.2.3: minor updates to the Connection Manager ASE services
- subclause 6.4: add Parameter ASE to the object table
- subclause 6.5: update contents of service table for Time Sync and Parameter ASEs

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

FDIS	Report on voting
65C/606/FDIS	65C/620/RVD

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

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

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

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

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

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

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

INTRODUCTION

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

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

Throughout the set of fieldbus standards, the term “service” refers to the abstract capability provided by one layer of the OSI Basic Reference Model to the layer immediately above. Thus, the application layer service defined in this standard 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 Overview

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

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 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 standard defines in an abstract way the externally visible service provided by the Type 2 fieldbus application layer in terms of

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

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

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

This standard 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 standard to provide access to the FAL to control certain aspects of its operation.

1.2 Specifications

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

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of services standardized as the various Types of IEC 61158, and the corresponding protocols standardized in subparts of IEC 61158-6.

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

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

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

1.3 Conformance

This standard 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 standard.

2 Normative references

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

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

IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

IEC/TR 61158-1:2010[1], *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-2:2007, *Industrial communication networks – Fieldbus specifications - Part 3-2: Data-link layer service definition – Type 2 elements*

IEC 61158-4-2:2010[1], *Industrial communication networks – Fieldbus specifications – Part 4-2: Data-link layer protocol specification – Type 2 elements*

IEC 61158-6-2:2010[1], *Industrial communication networks – Fieldbus specifications – Part 6-2: Application layer protocol specification – Type 2 elements*

1 To be published.

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

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

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

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

ISO/IEC 8859 (all parts), *Information technology – 8-bit single-byte coded graphic character sets*

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 Multiple-Octet Coded Character Set (UCS)*

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

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

ISO 11898:1993[2], *Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication*

IETF RFC 1759, *Printer MIB*, available at <http://www.ietf.org>

3 Terms, definitions, symbols, abbreviations and conventions

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

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

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

3.2 ISO/IEC 8822 terms

a) abstract syntax

b) presentation context

3.3 ISO/IEC 9545 terms

a) application-association

b) application-context

c) application context name

d) application-entity-invocation

e) application-entity-type

f) application-process-invocation

g) application-process-type

h) application-service-element

i) application control service element

3.4 ISO/IEC 8824 terms

a) object identifier

b) type

3.5 Type 2 fieldbus data-link layer terms

The following terms, defined in IEC 61158-3-2:2007 and IEC 61158-4-2:2010, 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

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

3.6.11
behavior
indication of how an object responds to particular events

3.6.12
boundary clock
clock with more than a single PTP port, with each PTP port providing access to a separate PTP communication path

NOTE Boundary clocks are used to eliminate fluctuations produced by routers and similar network elements.

3.6.13
class
a set of objects, all of which represent the same kind of system component

NOTE 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.

3.6.14
class attributes
attribute that is shared by all objects within the same class

3.6.15
class code
unique identifier assigned to each object class

3.6.16
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service

NOTE A class specific object is unique to the object class which defines it.

3.6.17
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

3.6.18
clock
device providing a measurement of the passage of time since a defined epoch

NOTE There are two types of clocks in IEC 61588:2009, boundary clocks and ordinary clocks.

3.6.19
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

3.6.20
connection
logical binding between application objects that may be within the same or different devices

NOTE 1 Connections may be either point-to-point or multipoint.

NOTE 2 The logical link between sink and source of attributes and services at different custom interfaces of RT-Auto ASEs is referred to as interconnection. There is a distinction between data and event interconnections. The logical link and the data flow between sink and source of automation data items is referred to as data interconnection. The logical link and the data flow between sink (method) and source (event) of operational services is referred to as event interconnection.

3.6.21
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

3.6.22
connection path
an octet stream that defines the application object to which a connection instance applies

3.6.23
connection point
buffer which is represented as a subinstance of an Assembly object

3.6.24
consume
act of receiving data from a producer

3.6.25
consumer
node or sink that is receiving data from a producer

3.6.26
consuming application
application that consumes data

3.6.27
cyclic
repetitive in a regular manner

3.6.28
device
physical hardware connected to the link

NOTE A device may contain more than one node.

3.6.29
device profile
a collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.6.30
end node
producing or consuming node

3.6.31
endpoint
one of the communicating entities involved in a connection

3.6.32
epoch
reference time defining the origin of a time scale
[IEC 61588:2009]

3.6.33
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.6.34
event
an instance of a change of conditions

3.6.35
frame
denigrated synonym for DLPDU

3.6.36
grandmaster clock
clock which serve as the primary source of time to which all others (within a collection of IEC 61588 clocks) are ultimately synchronized.

3.6.37
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

3.6.38
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

3.6.39
invocation
act of using a service or other resource of an application process

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

3.6.40
instance
the 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 The terms object, instance, and object instance are used to refer to a specific instance.

3.6.41
instance attributes
attribute that is unique to an object instance and not shared by the object class

3.6.42
instantiated
object that has been created in a device

3.6.43
logical device
a certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.6.44
Lpacket (or Link packet)
a piece of application information that contains a size, control octet, tag, and link data

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

3.6.45
management information
network-accessible information that supports managing the operation of the fieldbus system, including the application layer

NOTE Managing includes functions such as controlling, monitoring, and diagnosing.

3.6.46
master clock
single clock which serves as the primary source of time within each region, and which will in turn synchronize to other master clocks and ultimately to the grandmaster clock

NOTE A system of IEC 61588 clocks may be segmented into regions separated by boundary clocks.

3.6.47
member
piece of an attribute that is structured as an element of an array

3.6.48
Message Router
object within a node that distributes messaging requests to appropriate application objects

3.6.49
method
<object> a synonym for an operational service which is provided by the server ASE and invoked by a client

3.6.50
module
hardware or logical component of a physical device

3.6.51
multipoint connection
connection from one node to many

NOTE Multipoint connections allow messages from a single producer to be received by many consumer nodes.

3.6.52
network
a set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.6.53
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

3.6.54
object specific service
service unique to the object class which defines it

3.6.55
ordinary clock
IEC 61588 clock with a single PTP port

3.6.56
originator
client responsible for establishing a connection path to the target

3.6.57
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.6.58
physical device
an automation or other network device

3.6.59
point-to-point connection
connection that exists between exactly two application objects

3.6.60
Precision Time Protocol (PTP)
name used for the time synchronization protocol.

3.6.61
produce
act of sending data to be received by a consumer

3.6.62
producer
node that is responsible for sending data

3.6.63
property
general term for descriptive information about an object

3.6.64
PTP message
IEC 61588 time synchronization message

NOTE There are five designated messages types defined by IEC 61588:2009: Sync, Delay_Req, Follow-up, Delay_Resp, and Management.

3.6.65
PTP port
logical access point for IEC 61588 communications to the clock containing the port

3.6.66
resource
a processing or information capability of a subsystem

3.6.67
serial number
a unique 32-bit integer value assigned by each manufacturer/vendor to every device having Type 2 communication capabilities

NOTE The Vendor ID and serial number jointly form a unique identifier for each device.

3.6.68
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

3.6.69
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.6.70
synchronized clocks
(to a specified uncertainty) two clocks which have the same epoch and for which measurements of any time interval by both clocks differ by no more than the specified uncertainty

NOTE The timestamps generated by two synchronized clocks for the same event will differ by no more than the specified uncertainty.

3.6.71
System Time
absolute time value as defined by CPF2 time synchronization in the context of a distributed time system where all devices have a local clock that is synchronized with a common master clock

NOTE In the context of CPF2, System Time is a 64-bit integer value in units of nanoseconds with a value of 0 corresponding to the date 1972-01-01.

3.6.72
target
end-node to which a connection is established

3.6.73
Unconnected Message Manager (UCMM)
component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object

3.6.74
unconnected service
messaging service which does not rely on the set up of a connection between devices before allowing information exchanges

3.6.75
vendor ID
identification of each product manufacturer/vendor by a unique number

NOTE Vendor IDs are assigned by ODVA, Inc.

3.7 Type 2 abbreviations and symbols

AE	Application Entity
AL	Application layer
APO	Application object
AP	Application process
APDU	Application protocol data unit
AR	Application relationship
AREP	Application relationship end point
ASCII	American standard code for information interchange
ASE	Application service element
CID	Connection ID
CM_API	Actual packet interval
CM_RPI	Requested packet interval
Cnf	Confirmation
CR	Communication relationship
DL-	(as a prefix) data-link-
DLC	Data-link connection
DLCEP	Data-link connection end point
DLL	Data-link layer
DLM	Data-link-management
DLSAP	Data-link service access point
DLSDU	DL-service-data-unit
FAL	Fieldbus application layer
FIFO	First-in first-out

ID	Identifier	
IEC	International Electrotechnical Commission	
Ind	Indication	
IP	Internet protocol	
ISO	International Organization for Standardization	
I/O	Input/output	
LME	Layer management entity	
O2T	Originator to target (connection characteristics)	
O⇒T	Originator to target (connection characteristics)	
OSI	Open systems interconnect	
PDU	Protocol data unit	
PL	Physical layer	
PTP	Precision Time Protocol	[IEC 61588:2009]
QoS	Quality of service	
REP	Route endpoint	
Req	Request	
Rsp	Response	
SAP	Service access point	
SDU	Service data unit	
SEM	State event matrix	
STD	State transition diagram, used to describe object behavior	
T2O	Target to originator (connection characteristics)	
T⇒O	Target to originator (connection characteristics)	

3.8 Conventions

3.8.1 Overview

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of two parts, its class specification, and its service specification.

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

3.8.2 General conventions

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

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 standard, or by a user of this standard.

(3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the 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 standard 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 standard.

(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.

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

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

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

M	parameter is mandatory for the primitive
U	parameter is a User option, and may or may not be provided depending on dynamic usage of the service user. When not provided, a default value for the parameter is assumed.
C	parameter is conditional upon other parameters or upon the environment of the service user.
—	(blank) parameter is never present.
S	parameter is a selected item.

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

a) a parameter-specific constraint:

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

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

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

3.8.4.3 Service procedures

The procedures are defined in terms of

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

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

4 Common concepts

The common concepts and templates used to describe the application layer service in this standard are detailed in IEC/TR 61158-1:2010, 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/TR 61158-1:2010, 10.2.

5.2 Formal definition of data type objects

The template used to describe the data type class in this clause is detailed in IEC/TR 61158-1:2010, 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 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 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 ITIME

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.5.3 FTIME

CLASS: **Data type**

ATTRIBUTES:

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

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

5.3.1.5.4 LTIME

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

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

5.3.2 String types

5.3.2.1 BitString

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

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

5.3.2.2 OctetString

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

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet. IEC 61158-6-2:2010 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 61158-6-2:2010, 4.1.9.

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 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 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 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) contains 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, an

EPATH (Datatype_Element, limited to the values 0xD0, 0xD5, 0xD9, and0xDA) indicating the structure of the character string, a UINT (Charset_Element) indicating the character set which the character string is based on, and an array of octet elements which is the actual international character string. 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

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 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 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/TR 61158-1:2010, 9.7 are not used; instead they are specified in relevant Type-specific subclauses.
- there are no Abort or Reject services, only negative result confirmations, so the error procedure described in Annex A does not apply.

6.1.3 Relationships between ASEs

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

Object Management ASE:

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

Connection Manager ASE:

- Connection Manager (establishment and maintenance of connections)

or Connection ASE:

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

AR ASE:

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

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

Object Management ASE:

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

AR ASE:

— Transports (connected messaging and data services).

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

— ControlNet object (station management interfaces to lower layers, required in all devices using Type 2 data-link layer),

— Keeper object (holds and distributes attributes of all devices on the link, one required per link, with optional backup(s) – used in conjunction with Type 2 data-link layer),

— Scheduling object (holds information on link schedules, one required in each connection originator – used in conjunction with Type 2 data-link layer),

— TCP/IP Interface object (provides the mechanism to configure the TCP/IP interface of a device, required in all devices with a TCP/IP interface),

— Ethernet Link object (maintains link-specific counters and status information for an Ethernet port, required in all devices with an Ethernet port),

— DeviceNet object (station management interfaces to lower layers, required in all devices using ISO 11898:1993 data-link layer),

— Connection Configuration object (interface to create, configure and control connections in a device),

— DLR object (configuration and status information interface for the DLR protocol),

— QoS object (configuration interface for QoS-related behaviors in devices).

NOTE These lower layer management objects are described in IEC 61158-4-2:2010.

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

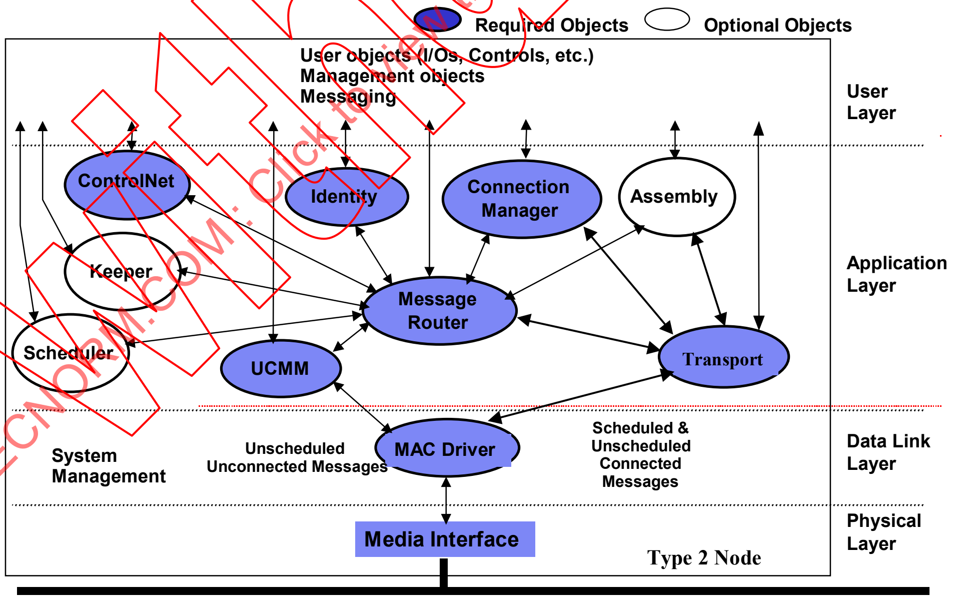


Figure 1 – Overview of ASEs and object classes

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.

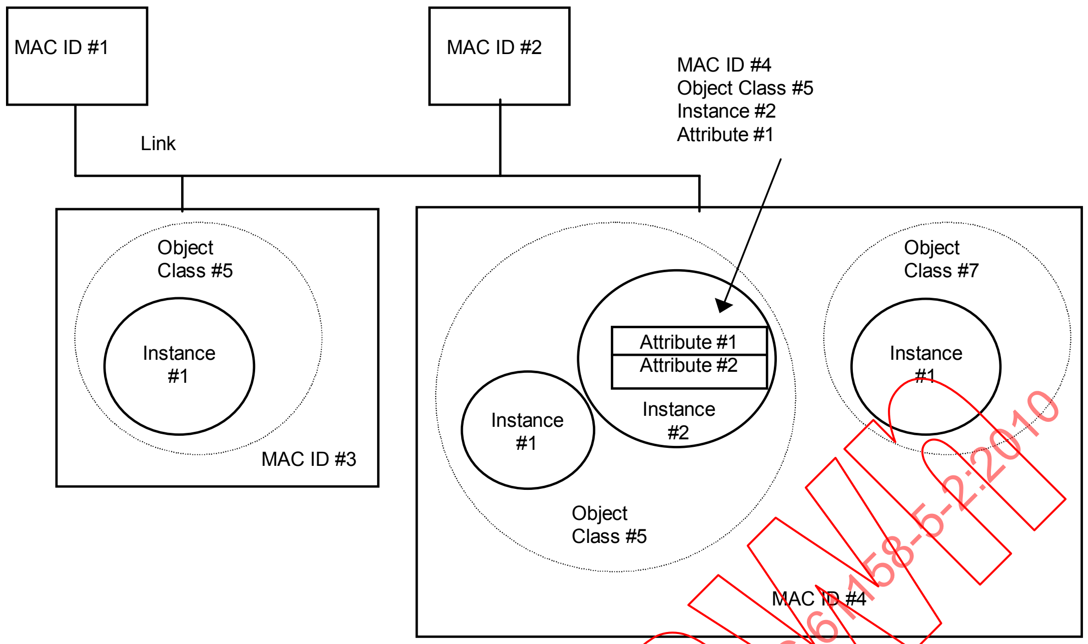


Figure 2 – Addressing format using MAC, class, instance and attribute IDs

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.

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

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

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

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

Information on how to address an object through multiple links or using a name is provided in IEC 61158-6-2:2010 (Path definition).

6.1.5 Data types

6.1.5.1 Data type general specification

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

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

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

The derived data types are the following:

directly derived;

enumerated;

subrange;

structured;

array.

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

6.1.5.2 Supported elementary data types

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

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

6.1.5.3 Compliance with IEC 61131-3

6.1.5.3.1 Compliance statement

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

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

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

Table 2 – Common elements

IEC 61131-3:2003 Table/feature	Feature description
10/1	BOOL data type
10/2	SINT data type
10/3	INT data type
10/4	DINT data type
10/5	LINT data type
10/6	USINT data type
10/7	UINT data type
10/8	UDINT data type
10/9	ULINT data type
10/10	REAL data type
10/11	LREAL data type
10/12	TIME data type
10/13	DATE data type
10/14	TIME_OF_DAY or TOD data type
10/15	DATE_AND_TIME or DT data type
10/16	STRING data type
10/17	SWORD data type
10/18	WORD data type
10/19	DWORD data type
10/20	LWORD data type
12/1	Directly derived data types
12/2	Enumerated data types
12/3	Subrange data types
12/4	Array data types
12/5	Structured data types
13	Standard default initial values
Subclause 2.5.1.3	User-declared functions (no table entry)
22/1	Type conversions (see Table 4)
22/2	TRUNC function
22/3	BCD_TO_** functions
22/4	*_TO_BCD functions
23/1-11	Standard functions of one numeric variable: ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN
24/12n-18n	Standard named arithmetic functions: ADD, MUL, SUB, DIV, MOD, EXPT, MOVE
24/ 12s-15s, 17s,18s	Standard symbolic arithmetic functions: +, *, –, /, **, :=
25/1-4	Standard bit string functions: SHL, SHR, ROR, ROL
26/5s-8s	Standard named bitwise Boolean functions: AND, OR, XOR, NOT
26/5s-7s	Standard symbolic bitwise Boolean functions: &, >=1, =2k+1
27/1-4	Standard selection functions: SEL, MAX, MIN, LIMIT, MUX
28/5n-10n	Standard named comparison functions: GT, GE, EQ, LE, LT, NE
28/5s-10s	Standard symbolic comparison functions: >, >=, =, <=, <, <>
29/1-9	Standard character string functions: LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND
30/1-14	Standard functions of time data types: (see NOTE) ADD, SUB, MUL, DIV, CONCAT, DATE_AND_TIME_TO_TIME_OF_DAY, TIME_OF_DAY_TO_DATE_AND_TIME
31/1-4	Standard functions of enumerated data types: SEL, MUX, EQ, NE
32	Standard access mechanisms to function block I/O parameters
33/8a,8b,9a,9b	User-defined function blocks per IEC 61131-3:2003, 2.5.2.2, with graphical or textual rising or falling edge input options
34/1-3	Standard bistable function blocks: SR, RS, SEMA
35/1,2	Standard edge detection function blocks: R_EDGE, F_EDGE
36/1-3	Standard counter function blocks: CTU, CTD, CTUD
37/1,2a,3a, 4	Standard timer function blocks: TP, TON, TOF, RTC
55/1-17	Standard operators: (), function evaluation, **,–, NOT, *, /, MOD, +,–, <, >, <=, >=, =, <>, &, AND, XOR, OR

NOTE IEC 61131-3:2003, Table 30, limits the data types to which these operations apply.

Table 3 – ST language elements

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

Table 4 – Type conversion operations

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

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

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

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

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

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

NOTE 5 Bit numbering in this expression is as specified in IEC 61158-6-2:2010.

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 IEC 60559.

NOTE 9 Rounding errors may 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 may 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 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 part are listed in Table 6. When these extensions are used in a particular device, the subclause references in this table shall be replaced by references to the descriptions of the corresponding extensions in the functional unit's documentation.

Table 6 – Extensions to IEC 61131-3:2003

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

6.2 ASEs

6.2.1 Object management ASE

6.2.1.1 Overview

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

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

– Settable (Set) – The attribute may be accessed by one of the Set_Attribute services (if the behavior of the device does not require a Set_Attribute service, then a device implementation of that object is not required to implement the attribute as settable);
– Settable attributes are also Gettable and may be accessed by Get_Attribute services;
– Gettable (Get) – The attribute may be accessed by one of the Get_Attribute services, but shall not be modified by any of the Set_Attribute services.

6.2.1.2 FAL management model class specification

6.2.1.2.1 General formal model

6.2.1.2.1.1 Class definition

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

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

FAL ASE:			**FAL Management ASE**
CLASS:			**Base_Object**
CLASS ID:			**NULL**
PARENT CLASS:			**TOP**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(*)	Attribute:	Revision
2	(o)	Attribute:	Max Instance
3	(o)	Attribute:	Number of Instances
4	(o)	Attribute:	Optional attribute list

4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):

1	(o)	Attribute:	Attr1

SYSTEM MANAGEMENT SERVICES:

1	(o)	Mgt Service:	Get_Attribute_All
2	(o)	Mgt Service:	Set_Attribute_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
10	(o)	Mgt Service:	Get_Attribute_Single
11	(o)	Mgt Service:	Set_Attribute_Single
12	(o)	Mgt Service:	Find_Next_Object_Instance
13	(o)	Mgt Service:	Apply_Attributes
14	(o)	Mgt Service:	Save
15	(o)	Mgt Service:	Restore
16	(o)	Ops Service:	Group_Sync

OBJECT MANAGEMENT SERVICES:

1	(o)	Ops Service:	Get_Attribute_All
2	(o)	Ops Service:	Set_Attribute_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:	Get_Attribute_Single
11	(o)	Ops Service:	Set_Attribute_Single
12	(o)	Ops Service:	NOP
13	(o)	Ops Service:	Apply_Attributes
14	(o)	Ops Service:	Save
15	(o)	Ops Service:	Restore
16	(o)	Ops Service:	Group_Sync

OBJECT SPECIFIC SERVICES:

1	(o)	OpsService:	Service1

6.2.1.2.1.2 Access attributes

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

Object Instance number

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

Symbolic name

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

6.2.1.2.1.3 System management attributes (class attributes)

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

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

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

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

Revision

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

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

Max Instance

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

Number of Instances

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

Optional attribute list

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

Number of attributes

Number of attributes in the optional attribute list.

Optional attributes

List of optional attribute numbers.

Optional service list

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

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

Number services

Number of services in the optional service list.

Optional services

List of optional service codes.

Maximum ID Number Class Attributes

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

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

Maximum ID Number Instance Attributes

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

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

6.2.1.2.1.4 Object management attributes

Attributes at the instance level

An Instance Attribute shall be unique to an object instance and not shared by the object class. 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.

Get_Attribute_All

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

Set_Attribute_All

The Set_Attribute_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.

Start

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

Stop

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

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.

Get_Attribute_Single

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

Set_Attribute_Single

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

Find_Next_Object_Instance

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

NOP

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

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.

Save

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

Restore

The Restore service restores the contents of an APO object class or instance attributes from a storage location accessible by the Save service.

Group_Sync

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

6.2.1.2.1.6 Object specific services

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

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

6.2.1.2.2 Identity formal model

6.2.1.2.2.1 Class definition

The Identity ASE provides identification and general information about the device. Instance one of the Identity object shall be present in all devices.

The first instance identifies the whole device. It shall be used for electronic keying and by applications wishing to determine what nodes are on the network. Other instances are optional. They may be provided by a device to give additional information about a device and its subsystems.

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

FAL ASE:	**FAL Management ASE**
CLASS:	**Identity_Object**
CLASS ID:	**1**
PARENT CLASS:	**Base_Object**

ACCESS ATTRIBUTES:

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

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):

1	(o)	Attribute:	Revision = 1
2	(*)	Attribute:	Max Instance
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):

1	(m)	Attribute:	Vendor ID
2	(m)	Attribute:	Device Type
3	(m)	Attribute:	Product Code
4	(m)	Attribute:	Revision
4.1	(m)	Attribute:	Major Revision
4.2	(m)	Attribute:	Minor Revision
5	(m)	Attribute:	Status
5.1	(m)	Attribute:	Owned
5.2	(m)	Attribute:	Configured
5.3	(m)	Attribute:	Additional 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
6	(m)	Attribute:	Serial Number
7	(m)	Attribute:	Product Name
8	(o)	Attribute:	State
9	(o)	Attribute:	Configuration Consistency Value
10	(o)	Attribute:	Heartbeat Interval
11	(o)	Attribute:	Active Language
12	(o)	Attribute:	Supported Language List
13	(o)	Attribute:	International Product Name
14	(o)	Attribute:	Semaphore
14.1	(m)	Attribute:	Client Electronic Key
14.2	(m)	Attribute:	Semaphore Timer
15	(o)	Attribute:	Assigned Name
16	(o)	Attribute:	Assigned Description
17	(o)	Attribute:	Geographic Location

SYSTEM MANAGEMENT SERVICES:

1	(o)	Mgt Service:	Get_Attribute_All
5	(o)	Mgt Service:	Reset
10	(o)	Mgt Service:	Get_Attribute_Single
12	(o)	Mgt Service:	Find_Next_Object_Instance

OBJECT MANAGEMENT SERVICES:

1	(m)	Ops Service:	Get_Attribute_All
5	(m)	Ops Service:	Reset
10	(o)	Ops Service:	Get_Attribute_Single

6.2.1.2.2.2 System management attributes (class attributes)

Max instance

If multiple subcomponents can be identified using instances of the Identity object, then this attribute is **mandatory**, else it is **optional**. The numerically lowest available integer shall be assigned as the Instance Identifier of a newly created Identity object instance.

6.2.1.2.2.3 Object management attributes

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

1: Vendor ID;

2: Device Type;

3: Product Code;

4: Revision.

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

Vendor ID

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

This instance attribute has an access rule of Get only.

Device type

Indication of general type of product. In order to allow interoperability and interchangeability, a Device Type shall be used to identify similar devices which exhibit the same behavior, produce and/or consume the same set of data, and contain the same set of configurable attributes. The formal definition of this information is known as a Device Profile: all devices with the same Device Type number shall meet the minimum requirements and implement the common options specified in the Device Profile for that device type. A listing of the Device Type ranges can be found in IEC 61158-6-2:2010.

This instance attribute has an access rule of Get only.

Product code

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

This instance attribute has an access rule of Get only.

Revision

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

The value zero is not valid for either the Major and Minor Revision fields of a compliant product. A damaged product may return zero to indicate unknown revision if its storage of revision become corrupt.

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

This 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 effect the configuration choices available to the user require incrementing the major revision.

Minor revision

The minor revision may be used to identify changes in a product that do not effect user configuration choices, e.g. bug fixes, hardware component change, labeling change. All firmware changes should, at a minimum, update the minor revision.

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 61158-6-2:2010.

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

This instance attribute has an access rule of Get only.

Owned

A (TRUE) indicates the device (or an object within the device) has an owner.

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.

Additional status

This attribute is used to provide more detailed information about device status, and can have one of the following values:

Self–Testing (power–up) or State Unknown

Firmware Update in progress

Communication Fault (lost connection)

Unkeyed, Awaiting Connection

Non–Volatile Configuration Bad

Major Fault

Connected, Active

Idle (program mode type)

Minor recoverable fault

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

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

Minor unrecoverable fault

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

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

Major recoverable fault

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

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

Major unrecoverable fault

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

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

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

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. Each manufacturer is responsible for guaranteeing the uniqueness of the serial number across all of its devices.

This instance attribute has an access rule of Get only.

Product name

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

This 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 may not be accurately reflected by the State attribute.

This instance attribute has an access rule of Get only.

Configuration consistency value

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

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

NOTE The client node’s behavior, upon detection of a mismatch, is vendor specific. The Configuration Consistency Value may 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 instance attribute has an access rule of Get only.

Heartbeat interval

Sets the nominal interval between production of optional heartbeat messages.

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

Supported language list

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

This instance attribute has an access rule of Get.

International product name

Names of the product in various languages.

This instance attribute has an access rule of Get.

Semaphore

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

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

Client electronic key

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

Semaphore timer

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

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

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

— The current value in the attribute is zero;

— The Electronic Key portion of the service data matches the Electronic Key portion of the attribute data.

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

Assigned name

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

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

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.

Reset

The Object Specific Data of the request primitive will contain a dedicated parameter to specify the type of Reset requested by the user. Its detailed format and corresponding Reset options are specified in IEC 61158-6-2:2010.

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

1. determine if it can provide the type of reset requested;
2. respond to the request;
3. attempt to perform the type of reset requested.

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

6.2.1.2.2.5 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 61158-4-2:2010 for details).

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

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

— the interval configured in the Heartbeat Interval Attribute has passed since the last heartbeat message;

— the Heartbeat message contents change, at a maximum rate of one “data changed” heartbeat message per second.

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

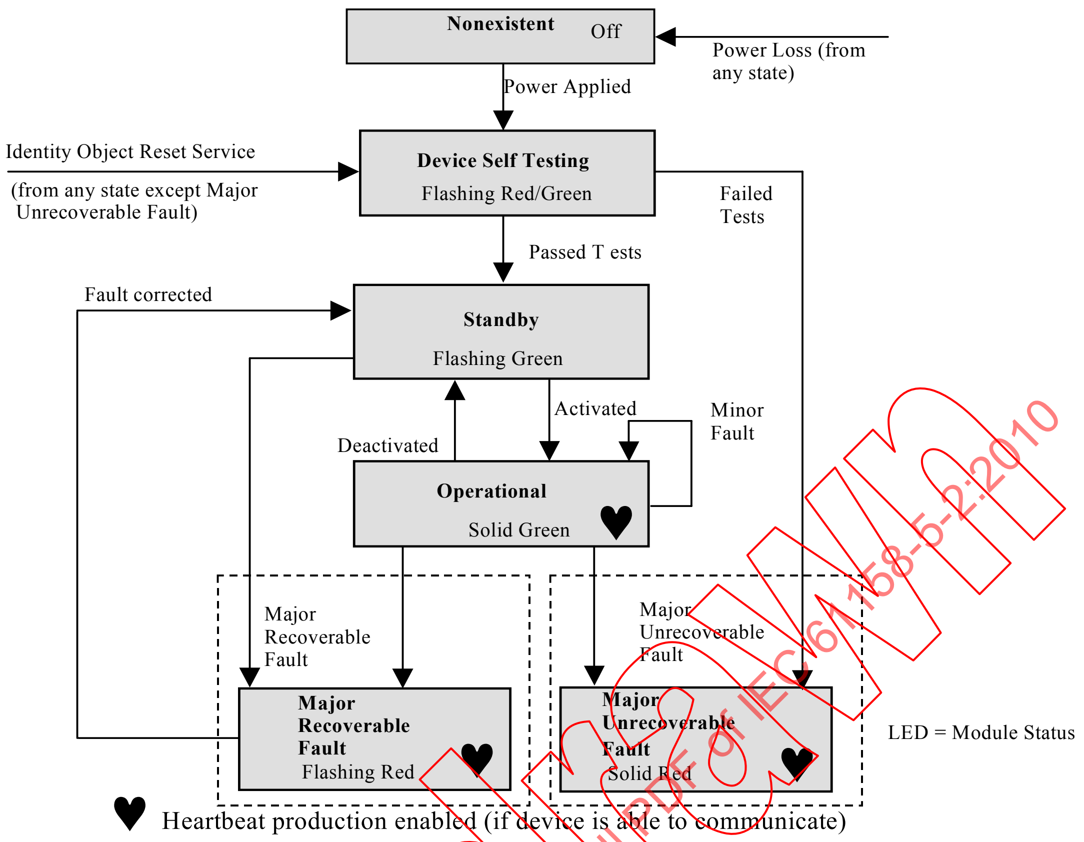


Figure 3 – Identity object state transition diagram

The STD for the Identity object contains the following events:

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

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.

6.2.1.2.3 Assembly formal model

6.2.1.2.3.1 Class definition

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

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

Assembly ASEs (objects) are either dynamic or static.

— Dynamic assemblies use assemblies member lists created and managed by the user. The member list may be altered by adding or deleting members;

— Static assemblies use member lists defined by the device profile or by the product implementers. The Instance number, number of members, and member list shall be fixed.

NOTE Static assemblies can usually be implemented entirely in ROM.

Instances of the Assembly object are divided into ranges specified as part of the device profiles.

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

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

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

FAL ASE:			**FAL Management ASE**
CLASS:			**Assembly_Object**
CLASS ID:			**4**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
3	(m)	Attribute:	Assembly type (STATIC, DYNAMIC)
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(m)	Attribute:	Revision = 2
2	(o)	Attribute:	Max Instance
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(*)	Attribute:	Number of Members in List
2	(*)	Attribute:	Member List
2.1	(m)	Attribute:	Member Data Size
2.2	(m)	Attribute:	Member Path Size
2.3	(m)	Attribute:	Member Path
3	(m)	Attribute:	Data

4	(o)	Attribute:	Size
SYSTEM MANAGEMENT SERVICES:			
8	(c)	Constraint:	Assembly type = DYNAMIC
8.1	(m)	Mgt Service:	Create
9	(c)	Constraint:	Assembly type = DYNAMIC
9.1	(o)	Mgt Service:	Delete
10	(m)	Mgt Service:	Get_Attribute_Single
OBJECT MANAGEMENT SERVICES:			
9	(c)	Constraint:	Assembly type = DYNAMIC
9.1	(m)	Ops Service:	Delete
10	(m)	Ops Service:	Get_Attribute_Single
11	(*)	Ops Service:	Set_Attribute_Single

6.2.1.2.3.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.1.2.3.3 Object management attributes

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

Number of members in list

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

Number of members in "Member List" attribute below.

This instance attribute has an access rule of Get only.

Member list

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

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

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

Member data size

Size of member data (in bits).

Member path size

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

Member path

Internal path to/from data for this member (packed format). See IEC 61158-6-2:2010 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 may contain many different data types. For efficiency it is best to keep this data word aligned by packing it on word boundaries and adding padding as needed. This can be accomplished by using “empty paths” (Member Path Size = 0).

This instance attribute has an access rule of Set.

Size

Number of octets in "Data" attribute.

This instance attribute has an access rule of Get only.

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

Delete

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

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, **mandatory** 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.

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 4 and Table 8 illustrate the behavior of Static Assembly objects.

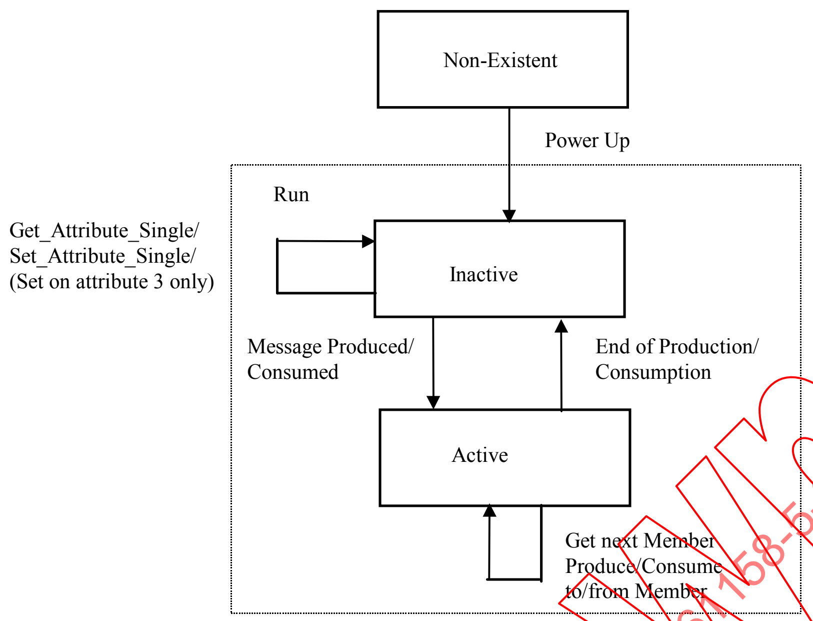


Figure 4 – Static Assembly state transition diagram

Table 8 – Static Assembly state event matrix

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

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

Dynamic Assembly state machine

Figure 5 and Table 9 illustrate the behavior of Dynamic Assembly objects.

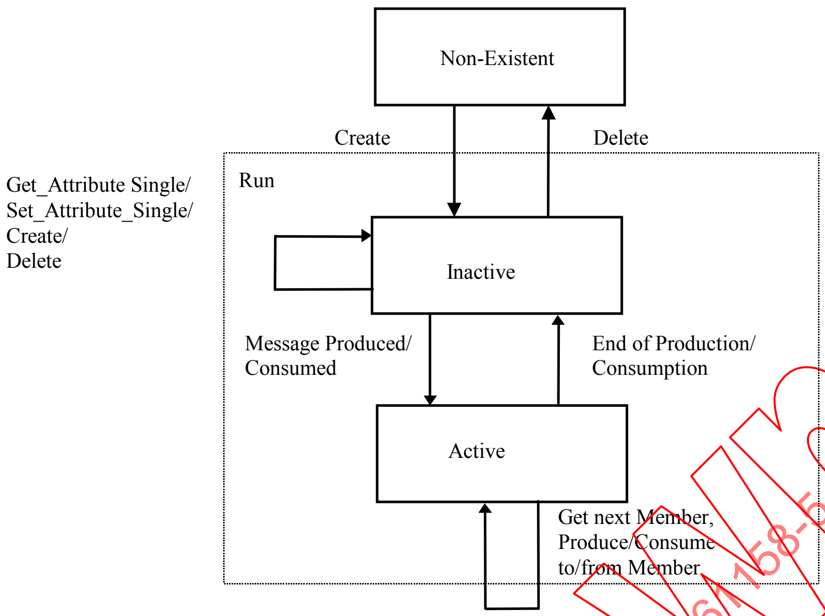


Figure 5 – Dynamic Assembly state transition diagram

Table 9 – Dynamic Assembly state event matrix

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

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

6.2.1.2.3.6 Specific requirements for a connection to the Assembly object

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

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

The format of the connection path shall be "class=Assembly, instance=XX, connection point=YY, connection point=ZZ" where XX is the instance, and YY/ZZ are the connection points. Any data segment in the path shall be used to set the data attribute of instance XX.

Connection point (instance) YY shall be the consumer (O⇒T) for the connection. Likewise, connection point (instance) ZZ shall be the producer (T⇒O).

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

– configuration,
– producer,
– consumer.

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

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

6.2.1.2.4 Message Router formal model

6.2.1.2.4.1 Class definition

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

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

The first part of the message (path information) shall then be parsed to determine the target object class. Interpretation of the class instance shall be performed on every service received by the Message Router. Any class instance 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. All connected service responses shall be routed back across the connection from which the service request was received. All unconnected service responses (UCMM) shall be returned via the same UCMM transaction that provided the request.

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

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

6.2.1.2.4.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.1.2.4.3 Object management attributes

Object_List

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

This instance attribute has an access rule of Get only.

Number

Number of supported classes in the Classes attribute below.

Classes

List of class codes supported by the device

Number available

Maximum number of connections supported by the Message Router

This instance attribute has an access rule of Get only.

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

Get_Attribute_Single

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

6.2.1.2.4.5 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 61158-6-2:2010 (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 10 shall also be supported.

Table 10 – Message Router object Forward_Open parameters

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

6.2.1.2.5 Acknowledge Handler formal model

6.2.1.2.5.1 Class definition

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

FAL ASE:			**FAL Management ASE**
CLASS:			**AckHandler_Object**
CLASS ID:			**43**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services

6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(m)	Attribute:	Acknowledge Timer
2	(m)	Attribute:	Retry Limit
3	(m)	Attribute:	COS Producing Connection Instance
4	(o)	Attribute:	Ack List Size
5	(o)	Attribute:	Ack List
6	(o)	Attribute:	Data with Ack Path List Size
7	(o)	Attribute:	Data with Ack Path List
SYSTEM MANAGEMENT SERVICES:			
8	(o)	Mgt Service:	Create
9	(o)	Mgt Service:	Delete
10	(o)	Mgt Service:	Get_Attribute_Single
OBJECT MANAGEMENT SERVICES:			
9	(o)	Ops Service:	Delete
10	(m)	Ops Service:	Get_Attribute_Single
11	(m)	Ops Service:	Set_Attribute_Single
OBJECT SPECIFIC SERVICES:			
1	(o)	Ops Service:	Add_AckData_Path
2	(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 millisecond 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 11 is the state event matrix for the Acknowledge Handler object associated with a Change of State connection. Table 12 is the state event matrix for the producing I/O application object.

Table 11 – Acknowledge Handler object state event matrix

Event	Acknowledge Handler object state		
	Non-existent	**Inactive**	**Active**
Receive Acknowledge	Not applicable	Ignore event	1) Clear Ack Flag 2) Forward any data to application object IF all acknowledges received: 3a) Clear Ack Timer and Retry Counter 3b) Send Acknowledge_Received event 3c) message to producing application object
Acknowledge Timer expires	Not applicable	Ignore event	Send Ack_Timeout event message to producing application object.
Data sent	Not applicable	Ignore event	1) Set Ack Required Flag 2) Set Ack Timer 3) Clear Retry Counter
Data resent	Not applicable	Ignore event	1) Set Ack Required Flag & Ack Timer IF Retry_Limit > 0 2a) Increment Retry Counter 2b) IF Retry Counter = Retry_Limit Send Rety_Limit_Reached event message to producing application object
Delete	Not applicable	Transition to Non-Existent	Transition to Non-Existent
Create	Transition to inactive	Not applicable	Not applicable
Apply attributes	Not applicable	Verify new connection can be added to list. Pass this message to the consumed application object, if one is configured for this connection.	Verify new connection can be added to list. Pass this message to the consumed application object, if one is configured for this connection.
Connection transition to Established	Not applicable	Add this connection instance to the connection list (or internally flag as 'Acking') . Pass this message to the consumed application object, if one is configured for this connection. Send Acknowledge_Active event message to producing application. Transition to Active.	Add this connection instance to the connection list. Pass this message to the consumed application object, if one is configured for this connection.
Inactivity/Watchdog Timer expires	Not applicable	Not applicable	1) Internally flag this connection as 'Not Acking'. An acknowledgement will no longer be monitored for this connection, however it remains in the connection list. 2) Pass this event to the consumed application object, if one is configured for this connection. 3) If no 'Acking' connections in list, send Acknowledge_Inactive event to producing application and transition to Inactive.
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 12 – 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
Acknowledge Inactive	Reset Ack Active Flag	Ignore event	1) Transition to Running 2) Reset Ack Active Flag	Reset Ack_Active Flag

Connection transition to Established	IF Ack Active Transition to Running with Acknowledg-ment
	IF Ack Inactive Transition to Running
		Ignore event	Ignore event	Ignore event

NOTE This is a partial state event matrix for a Producing I/O Application object. Only those states and events associated with data acknowledgement are defined. Other states and events will most likely be associated with a producing I/O application object.

6.2.1.2.5.7 Specific requirements for behavior and configuration

Behavior and configuration of acknowledged data production

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

Acknowledged data production

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

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

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

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

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

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

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

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

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

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

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

Use of timers with acknowledged data production

The following rules shall be observed when sending acknowledged data.

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

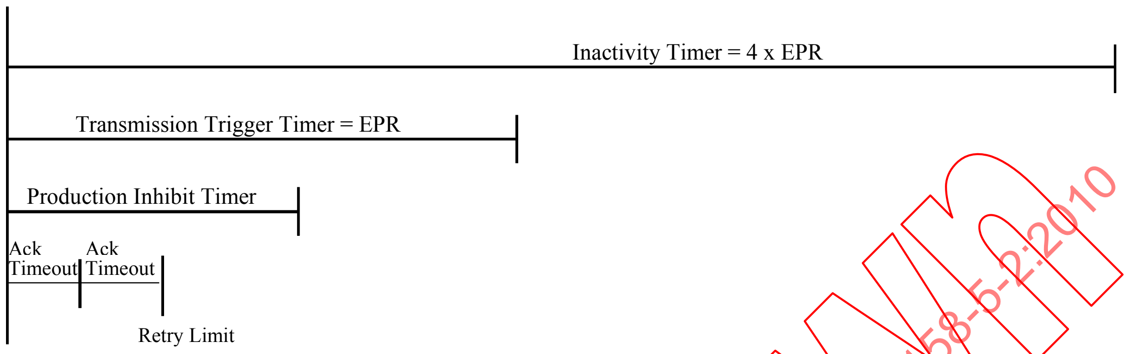


Figure 6 – Typical timing relationships for acknowledged data production

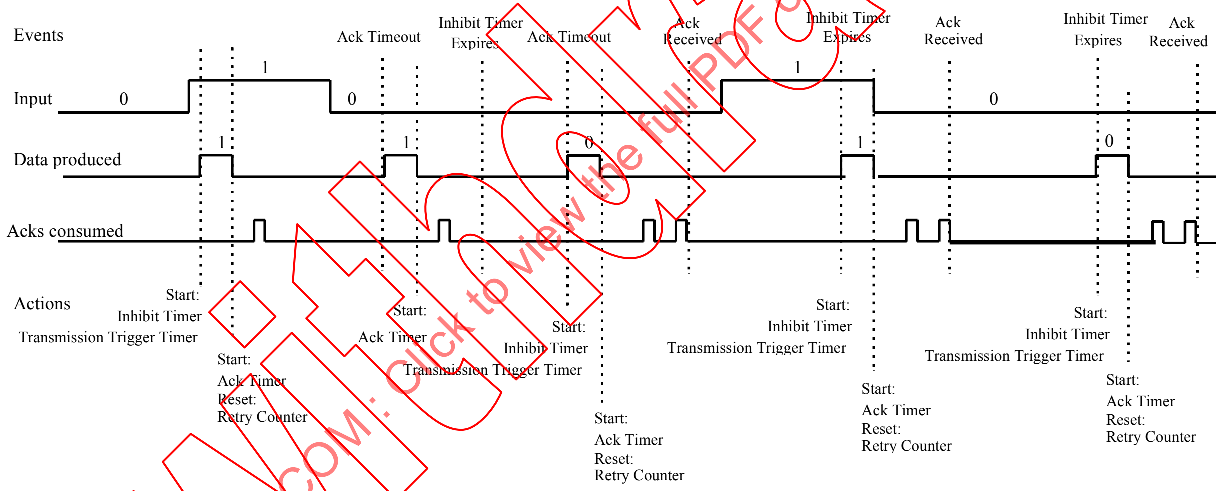

Events
Input
Data produced
Acks consumed
Actions
Ack Timeout
Inhibit Timer Expires
Ack Timeout
Ack Received
Inhibit Timer Expires
Ack Received
Inhibit Timer Expires
Ack Received
Start: Inhibit Timer Transmission Trigger Timer
Start: Ack Timer Reset: Retry Counter
Start: Ack Timer
Start: Inhibit Timer Transmission Trigger Timer
Start: Ack Timer Reset: Retry Counter
Start: Inhibit Timer Transmission Trigger Timer
Start: Ack Timer Reset: Retry Counter
Start: Inhibit Timer Transmission Trigger Timer
Start: Ack Timer Reset: Retry Counter

Figure 7 – 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 8) and multi-consumer configurations (see Figure 9).

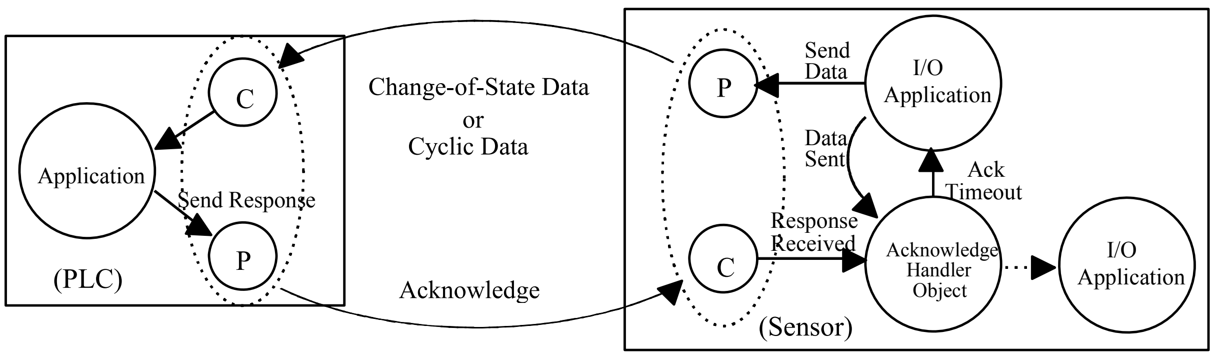


Figure 8 – Message flow in COS connection – one Connection object, one consumer

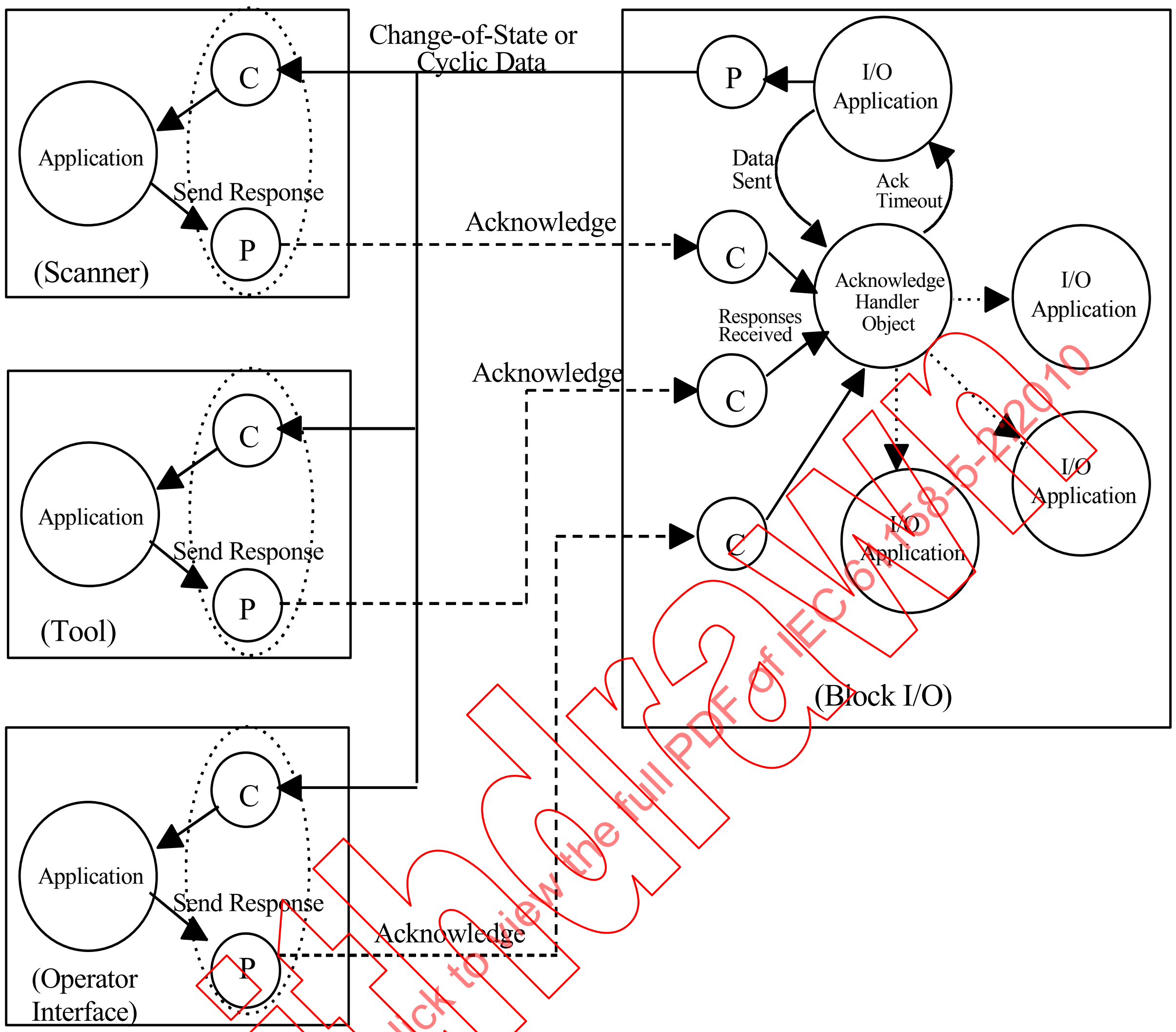


Figure 9 – Message flow in COS connection – multiple consumers

6.2.1.2.6 Time sync formal model

6.2.1.2.6.1 Class definition

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

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

This object provides attributes and services to:

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

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

FAL ASE:			**FAL Management ASE**
CLASS:			**Time_Sync_Object**
CLASS ID:			**67**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 2
2	(o)	Attribute:	Max Instance
3	(o)	Attribute:	Number of Instances
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
5	(o)	Attribute:	Optional service list
5.1	(m)	Attribute:	Number services
5.2	(m)	Attribute:	Optional services
6	(o)	Attribute:	Maximum ID Number Class Attributes
7	(o)	Attribute:	Maximum ID Number Instance Attributes
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(m)	Attribute:	PTPEnable
2	(m)	Attribute:	IsSynchronized
3	(m)	Attribute:	SystemTimeMicroseconds
4	(m)	Attribute:	SystemTimeNanoseconds
5	(m)	Attribute:	OffsetFromMaster
6	(m)	Attribute:	MaxOffsetFromMaster
7	(m)	Attribute:	MeanPathDelayToMaster
8	(m)	Attribute:	GrandMasterClockInfo
8.1	(m)	Attribute:	ClockIdentity
8.2	(m)	Attribute:	ClockClass
8.3	(m)	Attribute:	TimeAccuracy
8.4	(m)	Attribute:	OffsetScaledLogVariance
8.5	(m)	Attribute:	CurrentUtcOffset
8.6	(m)	Attribute:	TimePropertyFlags
8.7	(m)	Attribute:	TimeSource
8.8	(m)	Attribute:	Priority1
8.9	(m)	Attribute:	Priority2
9	(m)	Attribute:	ParentClockInfo
9.1	(m)	Attribute:	ClockIdentity
9.2	(m)	Attribute:	PortNumber
9.3	(m)	Attribute:	ObservedOffsetScaledLogVariance
9.4	(m)	Attribute:	ObservedPhaseChangeRate
10	(m)	Attribute:	LocalClockInfo
10.1	(m)	Attribute:	ClockIdentity
10.2	(m)	Attribute:	ClockClass
10.3	(m)	Attribute:	TimeAccuracy
10.4	(m)	Attribute:	OffsetScaledLogVariance
10.5	(m)	Attribute:	CurrentUtcOffset
10.6	(m)	Attribute:	TimePropertyFlags
10.7	(m)	Attribute:	TimeSource
11	(m)	Attribute:	NumberOfPorts
12	(m)	Attribute:	PortStateInfo
12.1	(m)	Attribute:	NumberOfPorts

12.2	(m)	Attribute:	PortNumber
12.3	(m)	Attribute:	PortState
13	(m)	Attribute:	PortEnableCfg
13.1	(m)	Attribute:	NumberOfPorts
13.2	(m)	Attribute:	PortNumber
13.3	(m)	Attribute:	PortState
14	(m)	Attribute:	PortLogAnnounceInervalCfg
14.1	(m)	Attribute:	NumberOfPorts
14.2	(m)	Attribute:	PortNumber
14.3	(m)	Attribute:	PortlogAnnounceInterval
15	(m)	Attribute:	PortLogSyncIntervalCfg
15.1	(m)	Attribute:	NumberOfPorts
15.2	(m)	Attribute:	PortNumber
15.3	(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	(m)	Attribute:	PortNumber
24.3	(m)	Attribute:	PortProfileIdentity
25	(m)	Attribute:	PortPhysicalAddressInfo
25.1	(m)	Attribute:	NumberOfPorts
25.2	(m)	Attribute:	PortNumber
25.3	(m)	Attribute:	PhysicalProtocol
25.4	(m)	Attribute:	SizeOfAddress
24.5	(m)	Attribute:	PortPhysiclAddress
26	(m)	Attribute:	PortProtocolAddressInfo
26.1	(m)	Attribute:	NumberOfPorts
26.2	(m)	Attribute:	PortNumber
26.3	(m)	Attribute:	NetworkProtocol
26.4	(m)	Attribute:	SizeOfAddress
26.5	(m)	Attribute:	PortProtocolAddress
27	(m)	Attribute:	StepsRemoved
28	(m)	Attribute:	SystemTimeAndOffset
28.1	(m)	Attribute:	SystemTime
28.2	(m)	Attribute:	SystemOffset

SYSTEM MANAGEMENT SERVICES:

10	(*)	Mgt Service:	Get_Attribute_Single

OBJECT MANAGEMENT SERVICES:

3	(m)	Mgt Service:	Get_Attribute_List
4	(m)	Mgt Service:	Set_Attribute_List

10	(m)	Ops Service:	Get_Attribute_Single
11	(m)	Ops Service:	Set_Attribute_Single

6.2.1.2.6.2 System management attributes (class attributes)

No specific requirement for this object.

6.2.1.2.6.3 Object management attributes

PTPEnable

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

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

IsSynchronized

Specifies whether the local clock is synchronized with a master reference clock. A clock is synchronized if it has one port in the slave state and is receiving updates from the time master. However, an application may impose more stringent synchronization requirements.

This instance attribute has an access rule of Get only.

SystemTimeMicroseconds

Specifies the current time in units of microseconds starting from the date 1970-01-01 at 00:00:00, Universal Coordinated Time (UTC). Leap seconds are distributed by the Precision Time Protocol as the current_utc_offset field.

PTP time and leap seconds are converted to microseconds to give the current System Time as: CurrentTime = PTPTime + LeapSeconds

See IEC 61588:2009 for a more detailed description of PTP time and the current_utc_offset field.

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

SystemTimeNanoseconds

Same as SystemTimeMicroseconds except in units of nanoseconds.

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

OffsetFromMaster

Specifies the amount of deviation between the local clock and its master clock in nanoseconds. See IEC 61588:2009 for more detail.

This 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. It may be set to 0.

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

MeanPathDelayToMaster

Specifies the average path delay between the local clock and master clock in nanoseconds. See IEC 61588:2009 for more detail.

This instance attribute has an access rule of Get only.

GrandmasterInfo, ParentInfo, LocalClockInfo

GrandmasterInfo, ParentInfo, and LocalClockInfo specify clock property information for the Grandmaster, Parent and Local PTP clock respectively. The data is extracted from the PTP data sets maintained by the PTP subsystem. The ClockIdentity provides a unique identifier for the clock. The clock time source, class, variance, and other attributes provide additional information about the properties of the clock.

These instance attributes have an access rule of Get only.

ClockIdentity

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

PortNumber

Specifies the port number of the port identity.

ClockClass

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

TimeAccuracy

Specifies the expected absolute accuracy of the clock relative to the PTP epoch. TimeAccuracy is the accuracy measure of clock quality used by the Best Master algorithm to determine the grandmaster. The accuracy is specified as a graduated scale starting at 25 nanoseconds and ending at greater than 10 seconds or unknown. A GPS time source will have an accuracy of approximately 250 nanoseconds. A Hand_Set clock will typically have accuracy less than 10 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 0 hours 1 January 2006 UTC, the offset was 33 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 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. Only a PTP Boundary clock will have more than one port. See IEC 61588:2009 for more detail of a PTP port. Also refer to the number_of_ports data member of the PTP default dataset.

This instance attribute has an access rule of Get only.

Port state

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

This instance attribute has an access rule of Get only.

PortEnableCfg

Specifies the port enable status of each port on the device. See IEC 61588:2009 for more detail. Also refer to the port_state data member of the PTP port dataset.

This 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 seconds.

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

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 seconds.

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

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.

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

This instance attribute has an access rule of Get only.

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. The value is between 0 and 255. The highest priority is 0.

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

This instance attribute has an access rule of Get only.

DomainNumber

Specifies the PTP clock domain.

ClockType

Specifies the type of the clock. A clock may implement more than one type of clock. A bit index set to one indicates the clock type is supported.

This instance attribute has an access rule of Get only.

ManufactureIdentity

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

This 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.

The format is UTF-8 Unicode. The maximum number of symbols is 64. The size field of the STRING data type is the total number of octets for the attribute.

This 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.

The format is UTF-8 Unicode. The maximum number of symbols is 32. The size field of the STRING data type is the total number of octets for the attribute.

This 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.

The format is UTF-8 Unicode. The maximum number of symbols is 128. The size field of the STRING data type is the total number of octets for the attribute.

This 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. The profile identity is contained in the first 6 octets of the array. The last two octets shall be set to zero.

This instance attribute has an access rule of Get only.

PortPhysicalAddressInfo

Specifies the physical protocol and physical address (e.g. ISO/IEC 8802-3) of each port of the device (e.g. MAC address). The maximum number of characters is 16. Unused array elements shall be set to zero.

This instance attribute has an access rule of Get only.

PortProtocolAddressInfo

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

This instance attribute has an access rule of Get only.

StepsRemoved

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

This instance attribute has an access rule of Get only.

SystemTimeAndOffset

Specifies the System Time in microseconds and the Offset to the local clock value. The responding device will return the current System Time and Offset.

This instance attribute has an access rule of Get only.

6.2.1.2.6.4 System management (class level) services

Get_Attribute_Single

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

6.2.1.2.6.5 Object management (instance level) services

Get_Attributes_List

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

Set_Attributes_List

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

Get_Attribute_Single

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

Set_Attribute_Single

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

6.2.1.2.6.6 Object specific services

No specific requirement for this object.

6.2.1.2.6.7 Specific behavior for the Time Sync object

System Time definition

The starting date/time for CPF2 time synchronization System Time is 1970-01-01 at 00:00:00. This is the starting epoch for both PTP and System Time. System Time is represented as a 64-bit value (ULINT) in nanoseconds or microseconds. 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. A negative System Time represents a time prior to the starting date/time.

In the Precision Time Protocol, time is distributed as a structure of type “TimeRepresentation”. This structure is made up of two 32-bit members. The first member of the structure represents the number of seconds since the beginning of the epoch (1970-01-01:00:00:00). The second member represents the number of nanoseconds. Leap seconds are distributed by the Precision Time Protocol as the current_utc_offset field.

PTP time and leap seconds are converted to microseconds or nanoseconds to give System Time as:

SystemTime = PTPTime – CurrentUtcOffset

Clock identity

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

The identifiers are formatted as specified in IEC 61158-6-2:2010, Table 89.

PTP port assignments

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

CPF 2 PTP profile

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

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

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

Profile identification

Table 13 identifies the CPF 2 PTP profile.

Table 13 – Profile identification

Profile identifier	Specification
PTP profile description	CIP Sync Profile
Version	1.0
Profile identifier	00-21-6C-00-01-00
Specifying organization	ODVA: ODVA Training and Technology Center 4220 Varsity Drive Suite A Ann Arbor, Michigan - USA 48108-5006 Phone: (1) 734-975-8840 Fax: (1) 734-922-0027 Email: conformance@odva.org Web: www.odva.org

CPF 2 PTP profile default settings and ranges

The CPF 2 profile shall use the PTP default and range settings as defined in Table 14.

Table 14 – Profile default settings and ranges

PTP Attribute	Attribute ID	Default	Range
Log announce interval	14	1	0 to 4
Log sync interval	15	0	-1 to +1
Priority1	16	128	0-255
Priority2	17	128	0-255
Domain	18	0	0-255
Announce receipt interval	Not exposed	3	2 to 10

PTP domain

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

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

PTP node management

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

Path delay mechanism

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

Recommended PTP clock settings

Table 15 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 15 – 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

TTL – Time to Live (CP 2/2 only)

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

Hand_Set clock quality management

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

Table 16 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 16 – Hand_Set clock quality management

Clock condition	Clock class	Clock accuracy	Time source
Immediately after hand setting.	Degradated Reference (B) (187)	Within 10 seconds (0x30)	Hand_Set (0x60)
Powerup out of the box	Default (248)	Unknown (0xFE)	Oscillator (0xA0)
NOTE In the table, a degraded reference B (187) is chosen 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.			

Clock model

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

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

These changes may occur due to one or more of the following conditions:

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

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

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

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

Figure 10 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.

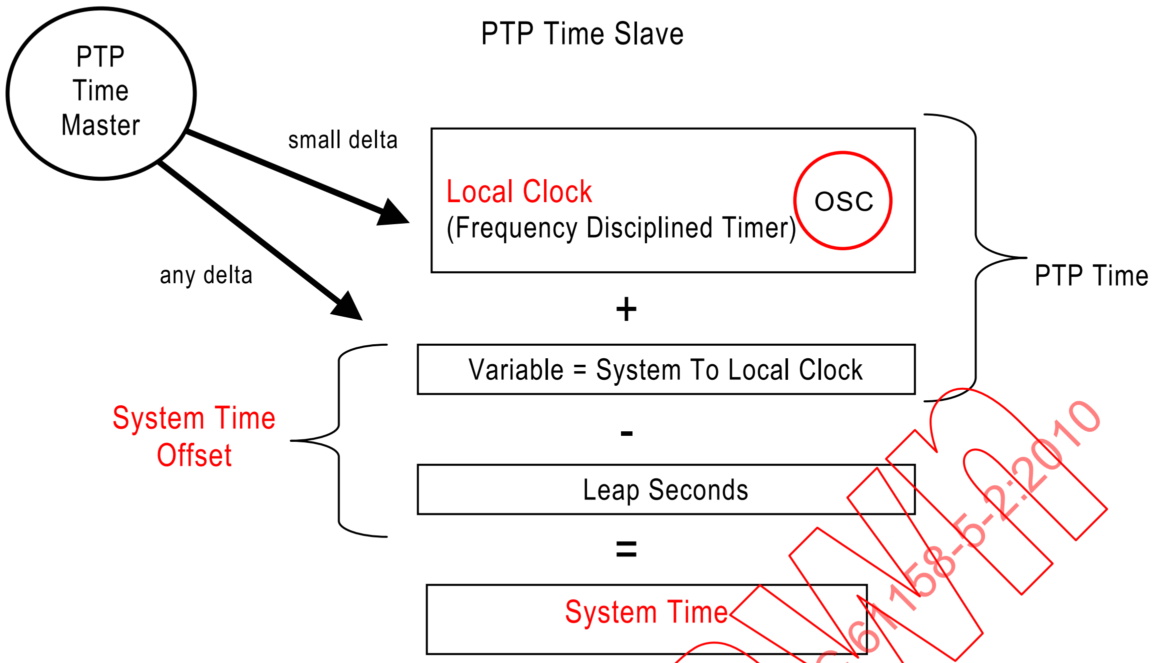


Figure 10 – CPF2 time synchronization offset clock model

Figure 11 shows a system of devices each implementing the CPF2 time synchronization Offset Clock Model. Each device is a part of a network of devices that share the same concept of System Time. Each device also has a Local Clock value based on a frequency disciplined timer and related to System Time via an offset value (System Time Offset). Such a system allows each device to maintain a local time independent from all other devices, but share a common notion of system time. As such, System Time may change without requiring changes to the local clock.

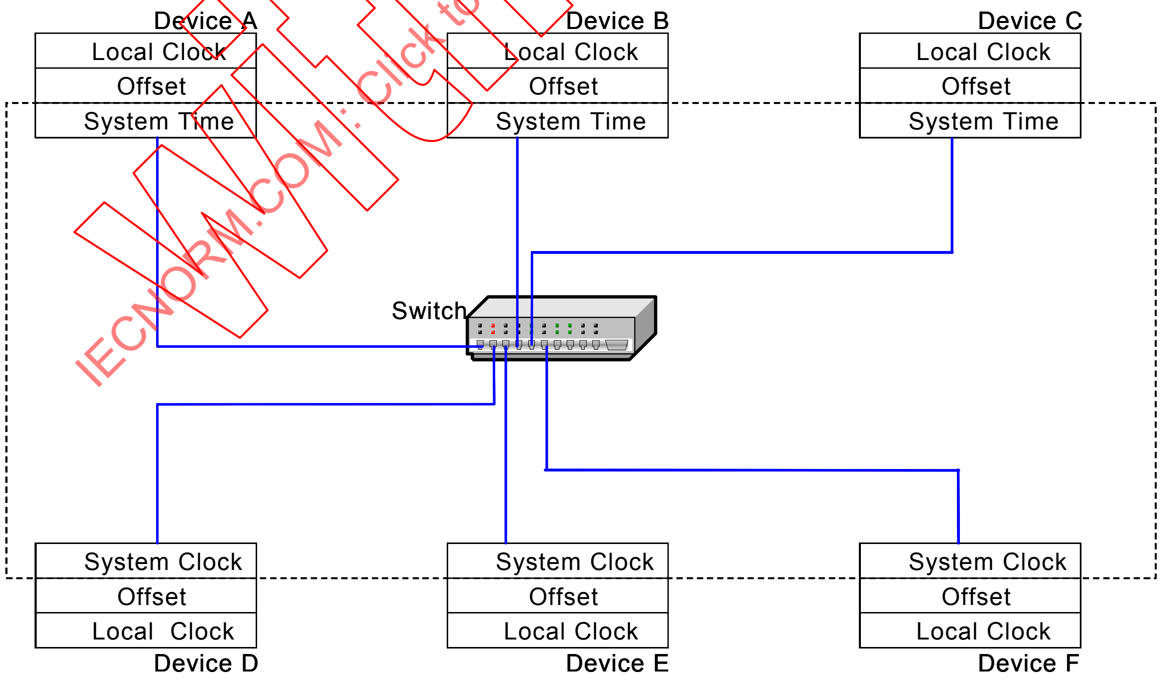


Figure 11 – CPF2 time synchronization system with offset clock model

System time compensation

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

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

Two possible algorithms are described in the following sections.

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

Step compensation algorithms

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

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

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:

$$\text{Timestamp}_C = \text{Timestamp}_0 + \text{Offset}_1 - \text{Offset}_0$$

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.

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:

$$\text{Timestamp}_{\text{Compensated}} = \text{Timestamp}_{\text{Received}} + ((\text{Dest}_{\text{Offset}} - \text{Dest}_{\text{LastOffset}}) - (\text{Source}_{\text{Offset}} - \text{Source}_{\text{LastOffset}}))$$

where:

$\text{Timestamp}_{\text{Received}}$ is received timestamp;

$\text{Dest}_{\text{Offset}}$ is the current value of the local clock time offset at the destination;

$\text{Dest}_{\text{LastOffset}}$ is the previous value of the local clock time offset at the destination;

$\text{Source}_{\text{Offset}}$ is the received value of the local clock time offset from the source;

$\text{Source}_{\text{LastOffset}}$ is the previous value of the local clock time offset from the source.

Last offsets are updated when:

$$|(\text{Dest}_{\text{Offset}} - \text{Dest}_{\text{LastOffset}}) - (\text{Source}_{\text{Offset}} - \text{Source}_{\text{LastOffset}})| <= \text{SyncBoundsLimit}$$

where:

SyncBoundsLimit is a relatively small number that defines that synchronization bounds for the application.

Group startup sequence

CPF2 time synchronization defines a group startup sequence and a group synchronizing service. The group startup sequence allows a group of devices to guarantee that their clocks have all been synchronized to the PTP master reference clock before starting an application that depends on System Time. The application defines the specific requirements needed for the group to be considered synchronized.

Each application (object) that wishes to synchronize to a group will implement the GroupSync 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 12.

Once the controller itself is synchronized to the master time clock, it initiates a series of GroupSync 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.

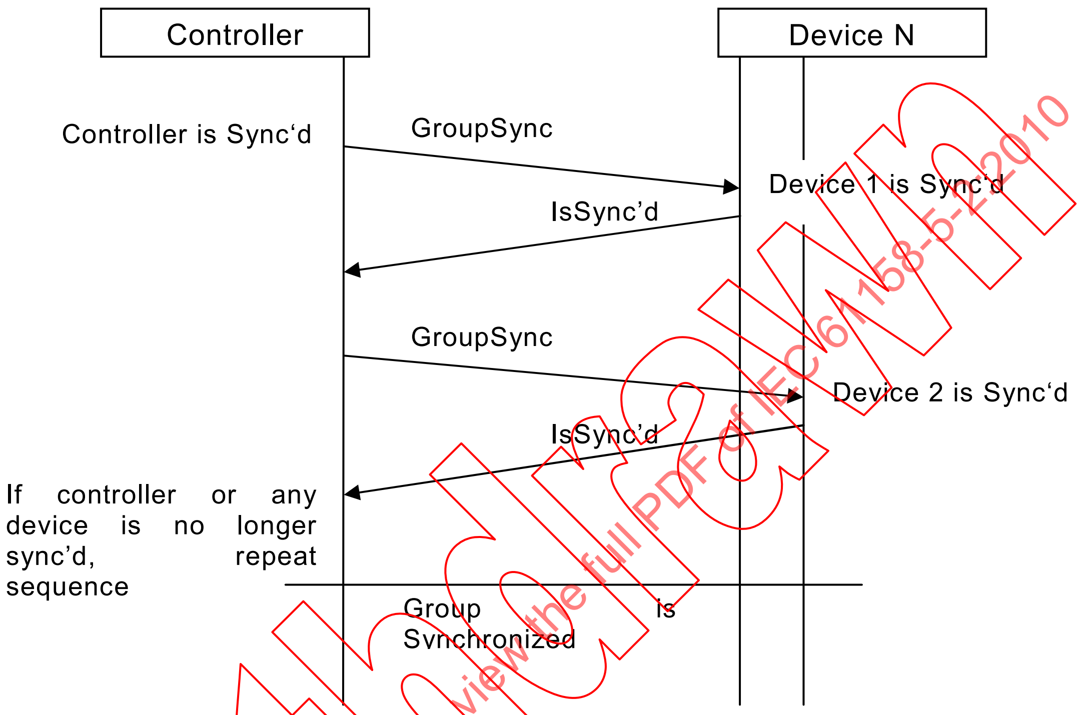


Figure 12 – CPF2 time synchronization group startup sequence

6.2.1.2.7 Parameter formal model

6.2.1.2.7.1 Class definition

Use of the Parameter object provides a known, public interface to a device's configuration data. In addition, this object also provides all the information necessary to define and describe each of a device's individual configuration parameters.

This object allows a device to fully identify a configurable parameter by supplying a full description of the parameter, including minimum and maximum values and a human–readable text string describing the parameter.

There shall be one instance of this object class for each of the device's configurable parameters. The instances shall start at instance one and increment by one with no gaps in the instances.

Each instance is linked to one of the configurable parameters, which is typically (but is not required to be) an attribute of one of the device's other objects. Changing the Parameter Value attribute of a Parameter object causes a corresponding change in the attribute value indicated by the Link Path attribute (the attribute value being pointed to by the Parameter object).

A Parameter Object Stub is a shorthand version of a Parameter object. The Stub stores only the configuration data value, providing only a standard access point for the parameter.

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

FAL ASE:			**FAL Management ASE**
CLASS:			**Parameter_Object**
CLASS ID:			**15**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
2	(o)	Attribute:	Max Instance
8	(m)	Attribute:	Parameter Class Descriptor
9	(m)	Attribute:	Configuration Assembly Instance
10	(o)	Attribute:	Native Language
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(m)	Attribute:	Parameter Value
2	(m)	Attribute:	Link path size
3	(m)	Attribute:	Link path
4	(o)	Attribute:	Description
5	(o)	Attribute:	Data Type
6	(o)	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	(*)	Mgt Service:	Get_Attributes_All
13	(o)	Mgt Service:	Apply_Attributes
14	(m)	Mgt Service:	Get_Attribute_Single
16	(m)	Mgt Service:	Set_Attribute_Single
21	(*)	Mgt Service:	Restore
22	(*)	Mgt Service:	Save
24	(*)	Mgt Service:	Get_Member

OBJECT SPECIFIC SERVICES:

75	(o)	Ops Service:	Get_Enum_String

6.2.1.2.7.2 System management attributes (class attributes)

Revision

The Revision level of this object.

This instance attribute has an access rule of Get.

Max Instance

Maximum instance number of an object currently created in this class level of the device.

This instance attribute has an access rule of Get.

Parameter Class Descriptor

Bits that describe parameters.

This instance attribute has an access rule of Get.

Configuration Assembly Instance

Instance number of the configuration assembly.

This instance attribute has an access rule of Get.

Native Language

Language ID for all character array accesses.

If the Active Language attribute of the Identity object (see 6.2.1.2.2) is supported, then this attribute shall not be supported.

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

6.2.1.2.7.3 Object management attributes

Parameter Value

Actual value of parameter. It can be read from or written to. This attribute is read–only if bit 4 of Attribute #4 is TRUE.

This instance attribute has an access rule of Get/Set. This attribute is used in the Stub and Full Parameter object.

Link path size

Size of link path.

This instance attribute has an access rule of Get/Set. This attribute is used in the Stub and Full Parameter object

Link path

CIP path to the object from where this parameter’s value is retrieved.

This instance attribute has an access rule of Get/Set. This attribute is used in the Stub and Full Parameter object

Description

Description of parameter.

This 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 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 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.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Units String

Engineering Unit String.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Help String

Help String.

This 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.

This 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.

This 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.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Multiplier

Multiplier for Scaling Factor.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Divisor

Divisor for Scaling Formula.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Base

Base for Scaling Formula.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Scaling Offset

Offset for Scaling Formula.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Multiplier Link

Parameter Instance of Multiplier source.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Divisor Link

Parameter Instance of Divisor source.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Base Link

Parameter Instance of Base source.

This instance attribute has an access rule of Get. This attribute is used in the Full Parameter object.

Offset Link

Parameter Instance of Offset source.

This 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.

This 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.

This instance attribute has an access rule of Get.

International Engineering Units

Engineering units associated with the parameter.

This instance attribute has an access rule of Get.

International Help String

Help String.

This instance attribute has an access rule of Get.

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 17 is the state event matrix for the Parameter object. The behavior of the Parameter object is illustrated in Figure 13.

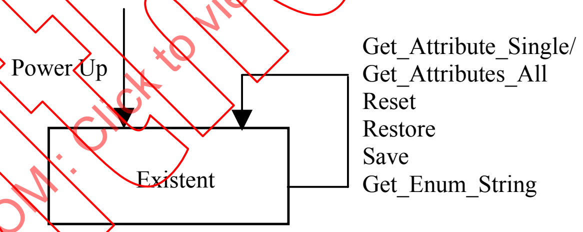


Figure 13 – Parameter object state transition diagram

Table 17 – 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

This subclause contains the definition of services that are unique to this ASE. The common services defined for this ASE are:

Get_Attribute_All

Set_Attribute_All

Get_Attribute_List

Set_Attribute_List

Reset

Start

Stop

Create

Delete

Get_Attribute_Single

Set_Attribute_Single

Find_Next_Object_Instance

NOP

Apply_Attributes

Save

Restore

Group_Sync

The object specific services defined for this ASE are:

Add_AckData_Path (Acknowledge Handler object)

Remove_AckData_Path (Acknowledge Handler object)

Get_Enum_String (Parameter object)

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 61158-6-2:2010 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. When used, the values submitted in the response primitive are delivered unchanged in the confirmation primitive.

6.2.1.3.3 FAL service status codes

Table 18 specifies the range of possible status codes.

Table 18 – Status codes

Name	Description and meaning of status code
Success	Service was successfully performed by the object specified
Connection failure	A connection related service failed along the connection path
Resource unavailable	Resources needed for the object to perform the requested service were unavailable
Invalid parameter value	See status code 0x20, which is the preferred value to use for this condition
Path segment error	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered
Path destination unknown	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered
Partial transfer	Only part of the expected data was transferred
Connection lost	The messaging connection was lost
Service not supported	The requested service was not implemented or was not defined for this class or object instance
Invalid attribute value	Invalid attribute data detected
Attribute list error	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non zero status
Already in requested mode/state	The object is already in the mode/state being requested by the service
Object state conflict	The object cannot perform the requested service in its current mode/state
Object already exists	The requested instance of object to be created already exists
Attribute not settable	A request to modify a non-modifiable attribute was received
Privilege violation	A permission/privilege check failed
Device state conflict	The device's current mode/state prohibits the execution of the requested service
Response data too large	The data to be transmitted in the response buffer is larger than the allocated response buffer
Fragmentation of a primitive value	The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type
Not enough data	The service did not supply enough data to perform the specified operation
Attribute not supported	The attribute specified in the request is not supported
Too much data	The service supplied more data than was expected
Object does not exist	The object specified does not exist in the device
Service fragmentation sequence not in progress	The fragmentation sequence for this service is not currently active for this data
No stored attribute data	The attribute data of this object was not saved prior to the requested service
Store operation failure	The attribute data of this object was not saved due to a failure during the attempt.
Routing failure, request packet too large for network	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 for network	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

Name	Description and meaning of status code
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 reply size). This status code can serve for other causes of invalid responses
Key Failure in path	The Key Segment which was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
Path Size Invalid	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
Unexpected attribute in list	An attempt was made to set an attribute that cannot be set at this time.
Invalid Member ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute.
Member not settable	A request to modify a non-modifiable member was received.
Group 2 only server general failure	This status code may only be reported by CP 2/3 Group 2 Only servers with 4 koctets or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable
Reserved for object class and service 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 61158-6-2:2010.

6.2.1.3.4 Get_Attribute_All

6.2.1.3.4.1 Service overview

The Get_Attribute_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 19.

Table 19 – Get_Attribute_All service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Data			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request. There are no specific parameters for this service.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute data

This parameter returns a stream of information containing all of the attributes. 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_Attribute_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.

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

No specific service procedure.

6.2.1.3.5 Set_Attribute_All

6.2.1.3.5.1 Service overview

The Set_Attribute_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 20.

Table 20 – Set_Attribute_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 all of the attributes. 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_Attribute_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.

Attributes shall be modified only if all attribute specific value verifications (e.g. range checks) are successful. If an error is detected, the Set_Attribute_All response shall return an appropriate error response message. If all verification checks pass, then the attributes shall be modified and a Set_Attribute_All success response shall be returned.

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 21.

Table 21 – Get_Attribute_List service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Count	M	M(=)		
Attribute List	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Count			M	M(=)
Attribute Data			M	M(=)
Attribute Identifier			M	M(=)
Attribute Status			M	M(=)
Attribute Value			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Attribute count

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

Attribute list

This parameter contains the list of the top-level attribute identifiers for the attributes that are being requested from the specified class or object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute count

This parameter indicates the number of attributes actually returned in the Attribute Data parameter below. This count can be different if the requested data does not fit entirely in the response buffer (see service procedure in 6.2.1.3.6.3).

Attribute data

This parameter returns a stream of information containing all or part of the requested attributes. Attributes shall be retrieved and packed in the sequence specified in the request.

Attribute identifier

This parameter contains the attribute identifier corresponding to the value being returned in Attribute Value.

Attribute status

This parameter indicates the individual status information for the requested attribute. It can either mirror one of the common service status codes, or be specific to this object/class attribute.

Attribute value

This parameter returns a stream of information containing all data for the requested attribute. Individual attribute data shall fit into the response buffer in its entirety (without fragmentation of individual attributes). The service shall access as many attributes as can fit into the response buffer.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Attribute list error

6.2.1.3.6.3 Service procedure

The attributes shall be specified using a list of attribute identifiers, which shall be supplied as a service parameter. If there is not enough space for an attribute's data in the response buffer, service processing shall terminate and the Attribute Count parameter shall be set to the number of attributes actually packed in the buffer. The client application (the requester), shall verify the value of the Attribute Count parameter in the response.

NOTE If necessary, the client application can submit another Get_Attribute_List service request for the attribute data remaining from its original list.

6.2.1.3.7 Set_Attribute_List

6.2.1.3.7.1 Service overview

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

6.2.1.3.7.2 Service primitives

The service parameters for this service are shown in Table 22.

Table 22 – Set_Attribute_List service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Attribute Count	M	M(=)		
Attribute Data	M	M(=)		
Attribute Identifier	M	M(=)		
Attribute Value	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Attribute Count			M	M(=)
Attribute Status List			M	M(=)
Attribute Identifier			M	M(=)
Attribute Status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Attribute count

This parameter indicates the number of attributes to be updated, and listed in the Attribute Data parameter below.

Attribute data

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

Attribute identifier

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

Attribute value

This parameter contains a stream of information containing all data for the target attribute. The data for an individual attribute shall be placed into the request buffer in its entirety (without fragmentation).

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute count

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

Attribute status list

This parameter returns a stream of information containing status information for each attribute listed in the service request.

Attribute identifier

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

Attribute status

This parameter indicates the individual status information for the specified attribute. It can either mirror one of the common service status codes, or be specific to this object/class attribute.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Attribute list error

Device state conflict

Missing attribute list entry data

6.2.1.3.7.3 Service procedure

The service shall be terminated if the response buffer is unable to accept the status of the next attribute.

Individual set operations shall not be interdependent (the request shall be treated as a series of independent set requests).

6.2.1.3.8 Reset

6.2.1.3.8.1 Service overview

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

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

6.2.1.3.8.2 Service primitives

The service parameters for this service are shown in Table 23.

Table 23 – 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 24.

Table 24 – 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 25.

Table 25 – Stop service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Object Specific Data	C	C(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Object Specific Data			C	C(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Object specific data

This conditional parameter, if specified, contains object class/instance specific parameters. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Already in requested mode/state

Object state conflict

Device state conflict

6.2.1.3.10.3 Service procedure

The effect of this service shall be documented in the individual object specification.

6.2.1.3.11 Create

6.2.1.3.11.1 Service overview

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

6.2.1.3.11.2 Service primitives

The service parameters for this service are shown in Table 26.

Table 26 – 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 27.

Table 27 – 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 of the specified object class (APO system management attributes) or instance (APO Object Management attributes).

6.2.1.3.13.2 Service primitives

The service parameters for this service are shown in Table 28.

Table 28 – 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.

Path

This parameter includes a path segment specifying the attribute number.

NOTE See IEC 61158-6-2:2010 for a description of path segments.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Attribute data

This parameter contains the requested attribute data.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Connection lost

Service not supported

Invalid attribute value

Attribute not settable

Device state conflict

Object does not exist

6.2.1.3.13.3 Service procedure

No specific service procedure.

6.2.1.3.14 Set_Attribute_Single

6.2.1.3.14.1 Service overview

The Set_Attribute_Single service modifies the contents of the specified attribute 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 29.

Table 29 – 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 61158-6-2:2010 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.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Object specific data

This conditional parameter, if specified, contains object class/instance specific information. If an object class/instance utilizes this field, then the object class/instance specification shall detail its format.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error among the following choices:

Path segment error

Path destination unknown

Connection lost

Service not supported

Invalid attribute value

Attribute not settable

Device state conflict

Object does not exist

6.2.1.3.14.3 Service procedure

No specific service procedure.

6.2.1.3.15 Find_Next_Object_Instance

6.2.1.3.15.1 Service overview

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

6.2.1.3.15.2 Service primitives

The service parameters for this service are shown in Table 30.

Table 30 – 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 14 provides a general example of this service.

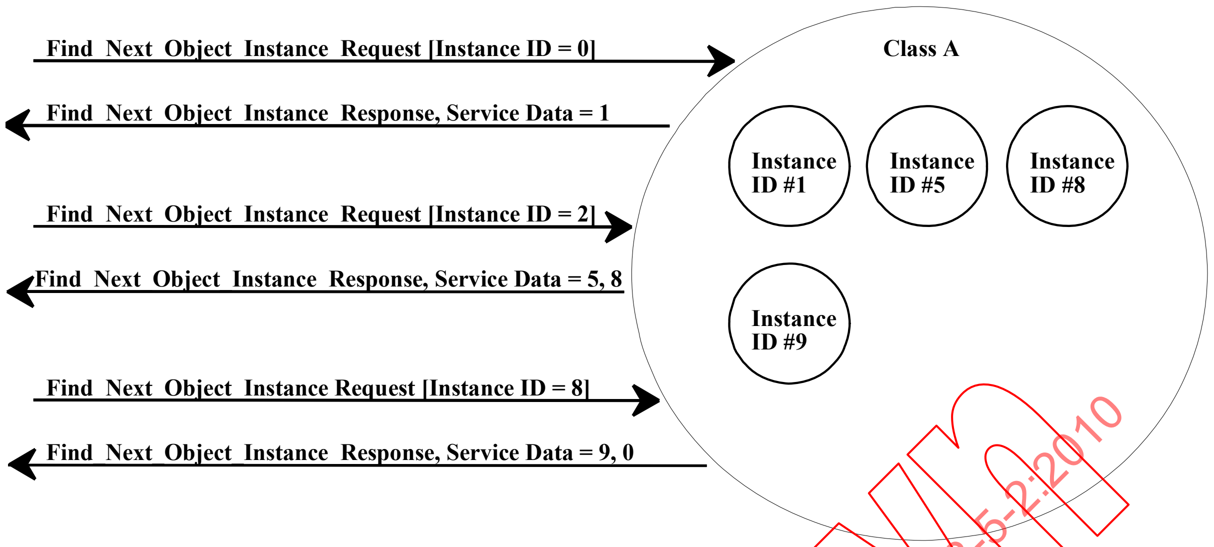

Find_Next_Object_Instance_Request [Instance ID = 0]
Find_Next_Object_Instance_Response, Service Data = 1
Find_Next_Object_Instance_Request [Instance ID = 2]
Find_Next_Object_Instance_Response, Service Data = 5, 8
Find_Next_Object_Instance Request [Instance ID = 8]
Find_Next_Object_Instance_Response, Service Data = 9, 0
Class A
Instance ID #1
Instance ID #5
Instance ID #8
Instance ID #9

Figure 14 – 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 may 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 31.

Table 31 – 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 32.

Table 32 – 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 33.

Table 33 – 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 34.

Table 34 – 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 Group_Sync

6.2.1.3.20.1 Service overview

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

6.2.1.3.20.2 Service primitives

The service parameters for this service are shown in Table 35.

Table 35 – 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 18).

6.2.1.3.20.3 Service procedure

The following list details requirements associated with the GroupSync service:

- The structure of the information in the request's Service Data Field adheres to the definition of the GroupSync 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 GroupSync 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 GroupSync service.

6.2.1.3.21 Add_AckData_Path

6.2.1.3.21.1 Service overview

The Add_AckData_Path adds a path for data with acknowledgment for a connected consumer.

6.2.1.3.21.2 Service primitives

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

Table 36 – 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 61158-6-2:2010 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 18)

6.2.1.3.21.3 Service procedure

No specific service procedure.

6.2.1.3.22 Remove_AckData_Path

6.2.1.3.22.1 Service overview

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

6.2.1.3.22.2 Service primitives

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

Table 37 – 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 18).

6.2.1.3.22.3 Service procedure

No specific service procedure.

6.2.1.3.23 Get-Enum_String

6.2.1.3.23.1 Service overview

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

6.2.1.3.23.2 Service primitives

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

Table 38 – 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 18).

6.2.1.3.23.3 Service procedure

No specific service procedure.

6.2.2 Connection manager ASE

6.2.2.1 Overview

The Connection Manager ASE (object) is used to manage the establishment and maintenance of communication connections. Every device shall implement instance 1 of the Connection Manager object.

6.2.2.2 Connection manager class specification

6.2.2.2.1 Connection manager formal model

6.2.2.2.1.1 Class definition

FAL ASE:			**FAL Management ASE**
CLASS:			**Connection_Manager_Object**
CLASS ID:			**6**
PARENT CLASS:			**Base_Object**
ACCESS ATTRIBUTES:			
1	(m)	Key Attribute:	Object Instance number
2	(o)	Key Attribute:	Symbolic name
SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):			
1	(o)	Attribute:	Revision = 1
2	(o)	Attribute:	Max Instance
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):			
1	(o)	Attribute:	OpenReqs
2	(o)	Attribute:	OpenFormat Rejects
3	(o)	Attribute:	OpenResource Rejects
4	(o)	Attribute:	OpenOther Rejects
5	(o)	Attribute:	CloseReqs
6	(o)	Attribute:	CloseFormat Rejects
7	(o)	Attribute:	CloseOther Rejects
8	(o)	Attribute:	ConnTimeouts

9	(o)	Attribute:	Connection Entry List
9.1	(m)	Attribute:	NumConnEntries
9.2	(m)	Attribute:	ConnOpenBits
11	(o)	Attribute:	CpuUtilization
12	(o)	Attribute:	MaxBuffSize
13	(o)	Attribute:	BufSize Remaining
SYSTEM MANAGEMENT SERVICES:			
1	(o)	Mgt Service:	Get_Attribute_All
3	(o)	Mgt Service:	Get_Attribute_List
10	(o)	Mgt Service:	Get_Attribute_Single
OBJECT MANAGEMENT SERVICES:			
1	(o)	Ops Service:	Get_Attribute_All
2	(o)	Ops Service:	Set_Attribute_All
3	(o)	Ops Service:	Get_Attribute_List
4	(o)	Ops Service:	Set_Attribute_List
10	(o)	Ops Service:	Get_Attribute_Single
11	(o)	Ops Service:	Set_Attribute_Single
OBJECT SPECIFIC SERVICES:			
1	(m)	OpsService:	CM_Open
2	(m)	OpsService:	CM_Close
3	(o)	OpsService:	CM_Unconnected_Send
4	(o)	OpsService:	CM_Get_Connection_Data
5	(o)	OpsService:	CM_Search_Connection_Data
6	(o)	OpsService:	CM_Get_Object_Owner

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 instance attribute has an access rule of Set.

OpenFormat rejects

Number of Open requests rejected by this node due to format errors.

This instance attribute has an access rule of Set.

OpenResource rejects

Number of Open requests rejected by this node.

This instance attribute has an access rule of Set.

OpenOther rejects

Number of Open requests rejected or timed out by downstream nodes.

This instance attribute has an access rule of Set.

CloseReqs

Number of Close requests received.

This instance attribute has an access rule of Set.

CloseFormat rejects

Number of Close requests rejected by this node due to format errors.

This instance attribute has an access rule of Set.

CloseOther rejects

Number of Close requests rejected or timed out by downstream nodes.

This 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 instance attribute has an access rule of Set.

Connection entry list

List of connections.

This 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 instance attribute has an access rule of Get only.

MaxBuffSize

Amount of buffer space originally available (buffer size is in octets).

This 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 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

No specific requirement for this object.

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_Object_Owner

This service is used to determine the owner of a redundant connection.

6.2.2.3 Connection manager ASE service specification

6.2.2.3.1 Supported services

This subclause 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_Object_Owner

6.2.2.3.2 Connection manager common service parameters

The following service parameters are common to all Connection Manager services.

NOTE 1 Complete specification of these parameters is provided in IEC 61158-6-2:2010.

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 61158-4-2:2010.

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. A value of MULTIPOINT indicates that other connections may later participate in the transport. Conversely, a value of POINT2POINT indicates that this connection shall not allow any other transport to participate.

Priority

The Priority parameter determines which data-link layer priority to use for the new transports, and shall be one of LOW, HIGH, or SCHEDULED.

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 actual packet interval (CM_API) specifies the time-out for the transport. If the application has not used the transport within this time-out, the transport shall close, which shall subsequently close the connection.

Trigger

The trigger parameter specifies the trigger mode, and shall be one of CYCLIC, CHANGE_OF_STATE, or APPLICATION.

NOTE 3 This parameter is used to configure the transport(s). Its meaning to the transports is described in 6.3.1.

Trans_class

The Trans_class parameter specifies the transport class selected, and shall be one of NULL, DUPLICATE_DETECTION, ACKNOWLEDGED, VERIFIED, NON-BLOCKING, NON-BLOCKING_FRAGMENTING, or MULTIPOINT_FRAGMENTING

NOTE 4 This parameter is used to configure the transport(s). Its meaning to the transports is described in 6.3.1.

Is_server

The Is_server parameter specifies if the new transport shall be of server (TRUE) or a client (FALSE).

NOTE 5 This parameter is used to configure the transport(s). Its meaning to the transports is described in 6.3.1.

Vendor_ID

Vendor ID of the device which has requested establishment of the connection (connection originator).

Originator_Ser_Num

Serial number of the device which has requested establishment of the connection (connection originator).

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.

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 61158-6-2:2010.

6.2.2.3.4 CM_Open

6.2.2.3.4.1 Service overview

The CM_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.

6.2.2.3.4.2 Service primitives

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

Table 39 – 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(=)		
Vendor_ID		M		
Originator_Ser_Num		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.

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_open confirm primitive shall be sent from the local originator Connection Manager back to the originating application. The confirmation shall indicate the status of the connection and shall provide information about the connection including the Transport Identifier parameter, which shall serve as a reference to the newly established connection.

6.2.2.3.5 CM_Close

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 40.

Table 40 – 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			
Vendor_ID		M		
Originator_Ser_Num		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. of a connection to de-establish the connection and de-allocate the resources associated with the connection. The service shall be sent from the originator application to the local Connection Manager.

The corresponding CM_close indication primitive shall then be sent from the target Connection Manager to the target application, after some local processing.

A successful CM_close request and response shall result in all non-shared resources associated with this connection in all nodes participating in the original connection being released, including connection IDs, bandwidth allocation, and internal memory buffers. Shared resources are those still in use by other connection(s) due to multicast. A router should also release non-shared resources if the response received indicates an error of General Status 0x01 and Extended Status 0x0107 (Target Connection Not Found).

For a CM_close service request received by a target node to be successful, it is sufficient that the Connection Triad matches an existing connection’s parameters. If there is no connection match, no connection is released and the target shall return an error with a General Status code of 0x01 and an Extended Status code of 0x0107 (Target Connection Not Found), unless the device detected an improper Connection Path. The Connection Path Size parameters may be ignored by the target node. However, an improperly formatted Connection_Path or Connection_Path mismatch may cause the service to be unsuccessful and return an error regardless of a Connection Triad match (see connection path error conditions below).

A CM_close service request received by an intermediate node along the path between the originator and target nodes shall be forwarded regardless of a Connection Triad match (in the case of a match not found, the connection may have timed out at this intermediate node). An improperly formatted Connection_Path shall, and a Connection_Path mismatch, may cause the service to be unsuccessful and return an error (see connection path error conditions below).

If either a target or intermediate node does detect a connection path error condition, it shall return an error response as described below. In all cases the connection remains unchanged (the connection is not released).

Improperly formatted Connection_Path:

- If a node detects that the value of the Connection_Path_Size field is smaller than the size of the Connection_Path, a General Status code of 0x15 shall be returned indicating that too much data is present in the service request.
- If a node detects that the value of the Connection_Path_Size field is greater than the size of the Connection_Path, General Status code 0x13 shall be returned indicating that not enough data is present in the service.

Connection_Path mismatch:

- If the node detects a mismatch in the Connection_Path between the value received in the CM_close request and the value sent in the originating connection request, a General Status code of 0x01 and an Extended Status code of either 0x316 (Error In Forward Close Service Connection Path Mismatch) or 0x0315 (Invalid Segment in Connection Path) shall be returned.

Success shall be returned when the connection has been deleted at the target. The originator, and each intermediate node along the path, closes the connection and releases resources associated with that connection when the success response is received.

6.2.2.3.5.4 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.

6.2.2.3.5.5 Service primitives

The service parameters for this service are shown in Table 41.

Table 41 – 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.

6.2.2.3.5.6 Service procedure

The CM_Unconnected_Send service shall use the Connection Manager object in each intermediate node to forward the message and to remember the return path. The UCMM of each link shall be used to forward the request from Connection Manager to Connection Manager just as it is for the Forward_Open service; however. no connection shall be built. The CM_Unconnected_Send service shall be sent to the local Connection Manager and shall be sent between intermediate nodes. When an intermediate node removes the last port segment, the message shall be formatted as a UCMM message and sent to the port and link address of the last segment.

NOTE The target node never sees the CM_Unconnected_Send service but only a standard message arriving via the UCMM.

The CM_Unconnected_Send response shall be generated by the last intermediate node from the UCMM response generated by the target node or by an intermediate node as the result of a UCMM time-out, a problem with the embedded message, or a problem with the Unconnected Service Request itself. The packet shall be routed from intermediate node to intermediate node using the information stored when the CM_Unconnected_Send request was processed. The response shall contain status information about the request and a response generated by the target node.

6.2.2.3.6 CM_Get_Connection_Data

6.2.2.3.6.1 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.

6.2.2.3.6.2 Service primitives

The service parameters for this service are shown in Table 42.

Table 42 – 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(=)
Vendor_ID			M	M(=)
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.

Serial_Number

This parameter contains the Serial Number of the target node.

6.2.2.3.6.3 Service procedure

No specific service procedure.

6.2.2.3.7 CM_Search_Connection_Data

6.2.2.3.7.1 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.

6.2.2.3.7.2 Service primitives

The service parameters for this service are shown in Table 43.

Table 43 – 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			
Vendor_ID	M	M(=)		
Serial Number	M	M(=)		
Connection Serial Number	M	M(=)		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Target transport identifier			M	M(=)
Connection state			M	M(=)
Originator port			M	M(=)
Target port			M	M(=)
Connection Serial Number			M	M(=)
Vendor_ID			M	M(=)
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.

Serial_Number

This parameter specifies the Serial Number of the target node.

Result

This parameter indicates whether the service request succeeded or failed. Its contents are identical to the Get_Connection_Data primitives.

6.2.2.3.7.3 Service procedure

No specific service procedure.

6.2.2.3.8 CM_Get_Object_Owner

6.2.2.3.8.1 Service overview

The CM_Get_Object_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.

6.2.2.3.8.2 Service primitives

The service parameters for this service are shown in Table 44.

Table 44 – 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_Ser_Num			M	M(=)

Argument

The argument contains the parameters of the service request.

Path

This parameter specifies the internal path from the Message Router in the target node to the selected object. It shall be the same path as would have appeared in a CM_Forward_Open request for this object, except that corresponding Additional Path electronic key, network, and data segments shall be removed before matching the paths.

Result

This parameter indicates whether the service request succeeded or failed.

Number of connections

This parameter contains the number of connections currently open to the specified path.

Number claiming ownership

This parameter contains the number of connections that are currently claiming ownership.

Number ready for ownership

This parameter contains the number of connections currently ready for ownership.

Last action

This parameter indicates mode of owning connection as follows. If the owning connection is in the idle mode, the Last action field shall equal 1. If the connection is in the run mode, the Last action field shall equal 2. If there is currently no owner, the Last action field shall equal 0. Values of the Last action field from 0x03 through 0xFE are reserved. If an implementation does not support one of these fields, the field shall equal 0xFF.

Connection serial number

Originator Vendor_ID

Originator_Ser_Num

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.

6.2.2.3.8.3 Service procedure

No specific service procedure.

6.2.3 Connection ASE

6.2.3.1 Overview

The Connection ASE (object) allocates and manages the internal resources associated with both I/O and Explicit Messaging Connections. The specific instance generated by the Connection ASE is referred to as a Connection instance or a Connection object.

A Connection object within a particular module actually represents one of the end-points of a connection. It is possible for one of the connection end-points to be configured and “active” (e.g. transmitting) without the other end-point(s) being present. Connection objects are used to model the communication specific characteristics of a particular application-to-application(s) relationship.

A specific Connection object Instance manages the communication specific aspects related to an end-point. A Connection object uses the services provided by a Link Producer and/or Link Consumer to perform low-level data transmission and reception functions.

6.2.3.2 Connection class specification

6.2.3.2.1 Connection formal model

6.2.3.2.1.1 Class definition

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

FAL ASE:	**Connection ASE**
CLASS:	**Connection _Object**
CLASS ID:	**5**
PARENT CLASS:	**Base_Object**

ACCESS ATTRIBUTES:

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

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES):

1	(o)	Attribute:	Revision = 1
2	(o)	Attribute:	Max Instance
4	(o)	Attribute:	Optional attribute list
4.1	(m)	Attribute:	Number of attributes
4.2	(m)	Attribute:	Optional attributes
8	(*)	Attribute	Connection Request Error Count
9	(*)	Attribute	Safety Connection Counters

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES):

1	(m)	Attribute:	State
2	(m)	Attribute:	Instance_type
3	(m)	Attribute:	TransportClass_trigger
4	(o)	Attribute:	CP2/3_produced_connection_id
5	(o)	Attribute:	CP2/3_consumed_connection_id
6	(o)	Attribute:	CP2/3_initial_comm_characteristics
7	(m)	Attribute:	Produced_connection_size
8	(m)	Attribute:	Consumed_connection_size
9	(m)	Attribute:	Expected_packet_rate
10	(o)	Attribute:	CPF2_produced_connection_id
11	(o)	Attribute:	CPF2_consumed_connection_id
12	(o)	Attribute:	Watchdog_timeout_action
13	(o)	Attribute:	Produced_connection_path_ length
14	(o)	Attribute:	Produced_connection_path
15	(o)	Attribute:	Consumed_connection_path_ length
16	(o)	Attribute:	Consumed_connection_path
17	(o)	Attribute:	Production_inhibit_time
18	(o)	Attribute:	Connection_timeout_multiplier
19	(o)	Attribute;	Connection_binding_list

SYSTEM MANAGEMENT SERVICES:

5	(o)	Mgt Service:	Reset
8	(o)	Mgt Service:	Create
9	(o)	Mgt Service:	Delete
10	(o)	Mgt Service:	Get_Attribute_Single
12	(o)	Mgt Service:	Find_Next_Object_Instance

OBJECT MANAGEMENT SERVICES:

5	(o)	Ops Service:	Reset
9	(o)	Ops Service:	Delete
10	(o)	Ops Service:	Get_Attribute_Single

11	(o)	Ops Service:	Set_Attribute_Single
13	(o)	Ops Service:	Apply_Attributes

OBJECT SPECIFIC SERVICES:

1	(o)	Ops Service:	Connection_Bind
2	(o)	Ops Service:	Producing_Application_Lookup
3	(*)	Ops Service:	SafetyOpen
4	(*)	Ops Service:	SafetyClose

6.2.3.2.1.2 System management attributes (class attributes)

Connection request error count

This attribute is used for safety (see IEC 61784-3-2).

Safety connection counters

This attribute is used for safety (see IEC 61784-3-2).

6.2.3.2.1.3 Object management attributes

Access rules for all Connection object management attributes are specified in 6.2.3.2.1.4.

State

State of the object.

Instance_type

Indicates either I/O or Messaging Connection.

TransportClass_trigger

Defines behavior of the Connection.

CP2/3_produced_connection_id

Placed in CAN Identifier field when the Connection transmits on a CP 2/3 subnet.

CP2/3_consumed_connection_id

CAN Identifier field value that denotes message to be received on a CP 2/3 subnet.

CP2/3_initial_comm_characteristics

Defines the Message Group(s) across which productions and consumptions associated with this Connection occur on a CP 2/3 subnet.

Produced_connection_size

Maximum number of octets transmitted across this Connection.

Consumed_connection_size

Maximum number of octets received across this Connection.

Expected_packet_rate

Defines timing associated with this Connection.

CPF2_produced_connection_id

Identifies the message sent on the subnet by this connection.

CPF2_consumed_connection_id

Identifies the message received from the subnet for this connection.

Watchdog_timeout_action

Defines how to handle Inactivity/Watchdog timeouts

Produced_connection_path_ length

Number of octets in the produced_connection_path attribute.

Produced_connection_path

Specifies the application object(s) whose data is to be produced by this Connection object.

Consumed_connection_path_ length

Number of octets in the consumed_connection_path attribute.

Consumed_connection_path

Specifies the application object(s) that are to receive the data consumed by this Connection object.

Production_inhibit_time

Defines minimum time between new data production. This attribute is required for all I/O Client connections, except those with a production trigger of Cyclic.

Connection_timeout_multiplier

Specifies the multiplier applied to the expected_packet_rate value to derive the value for the Inactivity/Watchdog Timer.

Connection_binding_list

List of I/O connection instances bound to this instance.

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

Connection object attribute access rules

During the configuration of a Connection instance using the Set_Attribute service, a module shall perform value checks of each separate attribute when it is modified. If an error is detected, then an Error Response is returned. The discovery of an error **DOES NOT** cause the deletion of the Connection instance. Table 45, Table 46 and Table 47 summarize the access rules for the different connection types

Important: If the produced_connection_id and/or consumed_connection_id attributes contain a non-default value upon reception of the Apply Request, then the related portion of the initial_comm_characteristics attribute is ignored and the ID value is validated and used.

The following series of tables indicates when an attribute can be read or written via a Get and/or Set operation based on the Connection's **state** and **instance_type.**

Important: If a request to Get/Set a supported attribute is received but the current state and/or instance_type of Connection dictates that the requested access is invalid, then the returned error status indicates: *The object cannot perform the requested service in its current mode/state*.

Table 45 – I/O Connection object attribute access

Attribute	I/O connection state				
	Non-existent	**Configuring**	**Waiting for connection ID**	**Established**	**Timed Out**
State	Not available	Get Only	Get Only	Get Only	Get Only
instance_type	Not available	Get Only	Get Only	Get Only	Get Only
transportClass_trigger	Not available	Get/Set	Get Only	Get Only	Get Only
produced_connection_id	Not available	Get/Set	Get/Set	Get/Set	Get/Set
consumed_connection_id	Not available	Get/Set	Get/Set	Get/Set	Get/Set
initial_comm_characteristics	Not available	Get/Set	Get Only	Get Only	Get Only
produced_connection_size	Not available	Get/Set	Get Only	Get Only	Get Only
consumed_connection_size	Not available	Get/Set	Get Only	Get Only	Get Only
expected_packet_rate[a]	Not available	Get/Set	Get Only	Get/Set	Get/Set
watchdog_timeout_action	Not available	Get/Set	Get Only	Get/Set	Get/Set
produced_connection_ path_length	Not available	Get Only	Get Only	Get Only	Get Only
produced_connection_path	Not available	Get/Set	Get Only	Get Only	Get Only
consumed_connection_ path_length	Not available	Get Only	Get Only	Get Only	Get Only
consumed_connection_path	Not available	Get/Set	Get Only	Get Only	Get Only
production_inhibit_time	Not available	Get/Set	Get Only	Get/Set	Get/Set
connection_timeout_multiplie r	Not available	Get/Set	Get Only	Get/Set [a]	Get/Set
Connection_binding_list	Not available	Get Only	Get Only	Get Only	Get Only

[a] When a Connection object is in the **Established** state, any modifications to the **expected_packet_rate** or **connection_timeout_multiplier** attributes have immediate effect on the Inactivity/Watchdog Timer. The following steps are performed by a Connection object in the **Established** state when a request is received to modify the **expected_packet_rate or connection_timeout_multiplier** attribute:

- the current Inactivity/Watchdog Timer is canceled,

- a new Inactivity/Watchdog Timer is activated based on the new value in the **expected_packet_rate** or **connection_timeout_multiplier** attribute.

Table 46 – Bridged Connection object attribute access

Attribute	Default value [a]	Bridged connection state			
		Non-existent	Configuring	Established	Closing
state	1	Not Available	Get Only	Get Only	Get Only
instance_type	2	Not Available	Get Only	Get Only	Get Only
transportClass_trigger	From service	Not Available	Get Only	Get Only	Get Only
produced_connection_id	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_id	From service	Not Available	Get Only	Get Only	Get Only
initial_comm_characteristics	From service	Not Available	Get Only	Get Only	Get Only
produced_connection_size	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_size	From service	Not Available	Get Only	Get Only	Get Only
expected_packet_rate	From service	Not Available	Get Only	Get Only	Get Only
watchdog_timeout_action	1	Not Available	Get Only	Get Only	Get Only
produced_connection_path_ length	From service	Not Available	Get Only	Get Only	Get Only
produced_connection_path	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_path_ length	From service	Not Available	Get Only	Get Only	Get Only
consumed_connection_path	From service	Not Available	Get Only	Get Only	Get Only
production_inhibit_time	From service	Not Available	Get Only	Get Only	Get Only
Connection_timeout_multipli er	From service	Not Available	Get Only	Get Only	Get Only
Connection_binding_list	Empty	Not Available	Get Only	Get Only	Get Only

[a] The default value is either the value indicated or is set based on one of the parameters of the Forward Open service received by the Connection Manager object.

Table 47 – Explicit messaging object attribute access

Attribute	Explicit messaging connection state	
	Non-existent	**Established/deferred delete**
State	Not available	Get Only
instance_type	Not available	Get Only
transportClass_trigger	Not available	Get Only
produced_connection_id	Not available	Get Only
consumed_connection_id	Not available	Get Only
initial_comm_characteristics	Not available	Get Only
produced_connection_size	Not available	Get Only
consumed_connection_size	Not available	Get Only
expected_packet_rate[1]	Not available	Get/Set [a]
watchdog_timeout_action	Not available	Get/Set
produced_connection_ path_length	Not available	Get Only
produced_connection_path	Not available	Get Only
consumed_connection_ path_length	Not available	Get Only
consumed_connection_path	Not available	Get Only
production_inhibit_time	Not available	Get Only
connection_timeout_multiplier	Not available	Get/Set [a]
Connection_binding_list	Not available	Get Only

[a] When a Connection object is in the Established state, any modifications to the expected_packet_rate or connection_timeout_multiplier attributes have immediate effect on the Inactivity/Watchdog Timer.

The following steps are performed by a Connection object in the Established state when a request is received to modify the expected_packet_rate or connection_timeout_multiplier attribute:
– the current Inactivity/Watchdog Timer is cancelled,
– a new Inactivity/Watchdog Timer is activated based on the new value in the expected_packet_rate or connection_timeout_multiplier attribute.

6.2.3.2.1.5 Object specific services

Connection_Bind

The Connection_Bind object specific service binds two dynamically created I/O connection instances together for purposes of connection timeouts and deletions.

Producing_Application_Lookup

The Producing_Application_Lookup object specific service provides a mechanism to find one or more connection instances in the Established state producing data from a given application object.

SafetyOpen

See IEC 61784-3-2 for the definition of this service.

SafetyClose

See IEC 61784-3-2 for the definition of this service.

6.2.3.2.1.6 Connection object state machines

See IEC 61158-6-2:2010, Clause 7 (AP-Context state machine).

6.2.3.2.1.7 Specific requirements of the Connection object

Connection Timing

Three *types* of timers are involved in a connection:

- Transmission Trigger Timer
- Inactivity/Watchdog Timer
- Production Inhibit Timer

The first two timers are initialized based on the value in the **expected_packet_rate** attribute.

Important: For Explicit Messaging, the Application is responsible for providing *response timeout* facilities. The amount of time a Client waits for a Server to respond to a request depends on the application and, possibly, on the service. Due to Media Access mechanisms used for accessing the subnet, an implementation should wait until an Explicit Request Message is transmitted before activating any response related timers.

Transmission trigger timer

This timer is required to be managed by the application within the **Client end-point** of a Connection. Expiration of this timer is an indication that the associated Connection object may need to be told to transmit a message. If a production has not occurred since the timer was activated, then the Connection object should be told to produce to avoid an Inactivity/Watchdog timeout at the Server end-point(s).

The tasks listed below are performed by a Connection immediately upon producing a message:

- the current value of the Transmission Trigger Timer is restored to its initial value and the timer is stopped;
- a new Transmission Trigger Timer is activated.

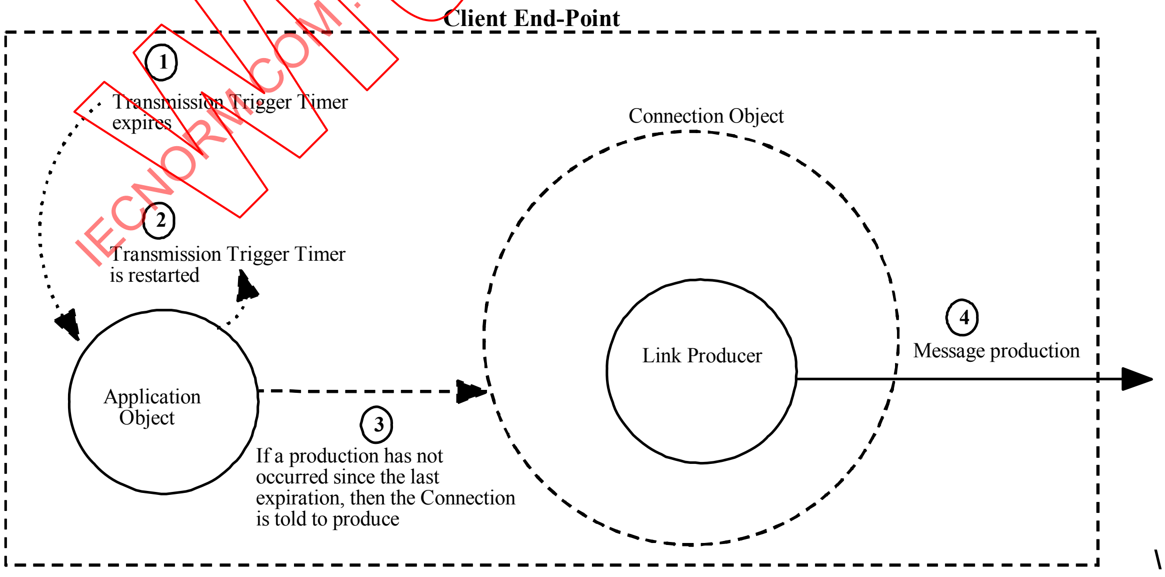

Client End-Point
1
Transmission Trigger Timer expires
2
Transmission Trigger Timer is restarted
Application Object
3
If a production has not occurred since the last expiration, then the Connection is told to produce
Connection Object
Link Producer
4
Message production

Figure 15 – Transmission trigger timer

As illustrated in Figure 15 when the Transmission Trigger Timer expires, it is immediately re-started. This timer is activated when the Connection transitions to the **Established** state.

Important: The Transmit Trigger Timer is initialized with the value in the expected_packet_rate attribute. If the expected_packet_rate attribute specifies the value zero (0), then the Transmission Trigger Timer is not activated and/or used by the Client end-point.

Important: Server end-points do not activate this timer.

Inactivity/watchdog timer

This timer is required to be managed by any **consuming** Connection object. Consuming Connection objects include:

- Client end-point Connection objects whose transportClass_trigger attribute indicates either Transport Class 2 or 3.
- All Server end-point Connection objects.

This timer is activated when the Connection transitions to the **Established** state. The tasks listed below are performed by a Connection immediately upon detecting that a valid message has been consumed:

- The current value for the Inactivity/Watchdog Timer is restored to its initial value and the timer is stopped.
- A new Inactivity/Watchdog Timer is activated.

The bullet items above indicate that the new Inactivity/Watchdog Timer is activated before the received message is processed.

Expiration of this timer is an indication that the Connection object has timed out while waiting to consume (see Figure 16). The Connection object performs the following steps when the Inactivity/Watchdog timer expires:

- issues an indication of this event to the application,
- performs the action indicated by the watchdog_timeout_action attribute.

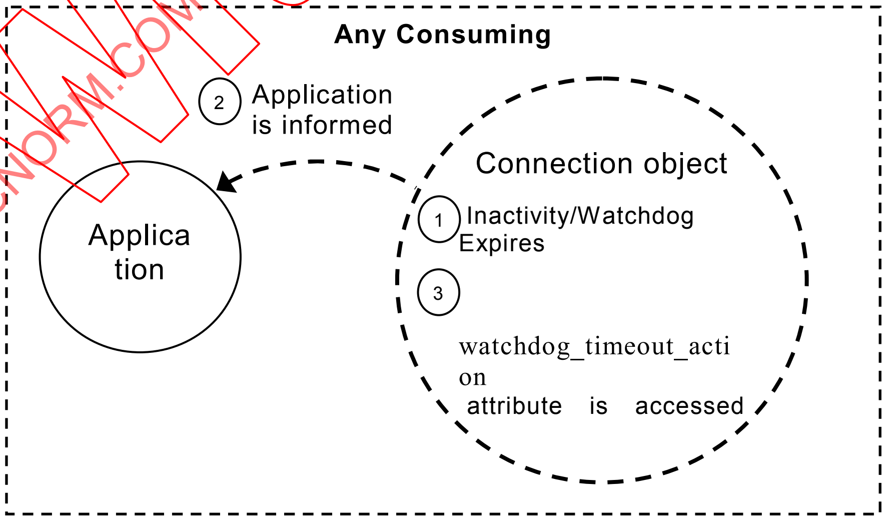

Any Consuming
2
Application is informed
Applica tion
Connection object
1
Inactivity/Watchdog Expires
3
watchdog_timeout_action attribute is accessed

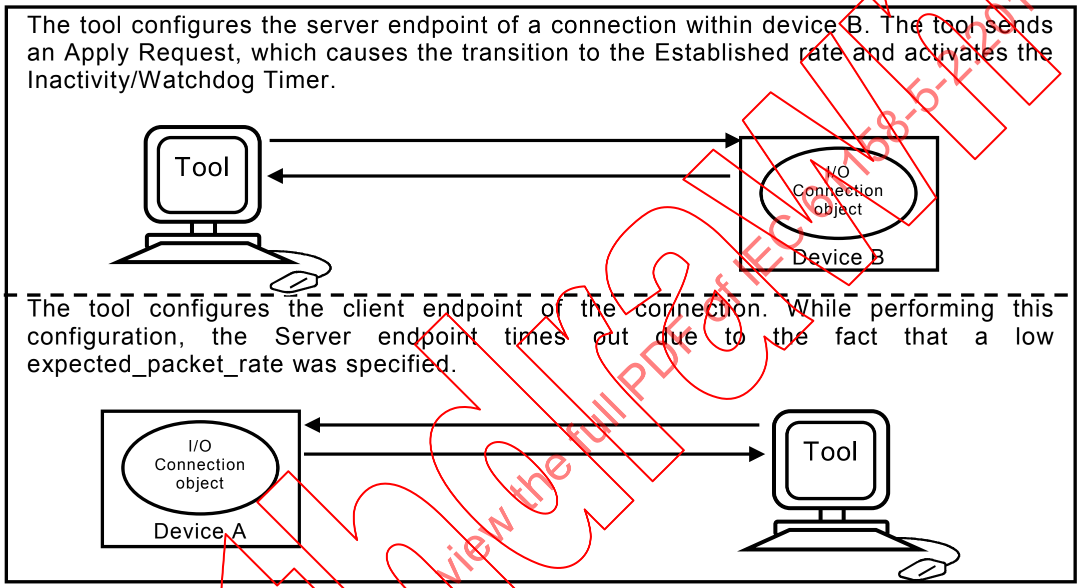
Figure 16 – Inactivity watchdog timer

Two different values are used for the Inactivity/Watchdog Timer, based on whether or not the Connection object has consumed a message:

1. The initial value loaded into the Inactivity/Watchdog Timer is either 10 000 milliseconds (10 seconds) or the **expected_packet_rate** multiplied by the **connection_timeout_multiplier**, depending on which value is numerically greater.

NOTE 1 If the **expected_packet_rate** attribute specifies the value zero (0), then the Inactivity/Watchdog Timer is not activated and/or used by the Connection object. A Connection object whose **expected_packet_rate** attribute is zero (0) will never experience an Inactivity/Watchdog timeout.

If the **expected_packet_rate** attribute multiplied by the **connection_timeout_multiplier** is greater than 10 000, then the **expected_packet_rate** multiplied by the **connection_timeout_multiplier** is used. Otherwise, 10 000 (10 seconds) is used. This is referred to as the **pre-consumption** timeout value. This value is used because a Connection may transition to the **Established** state before all end-points are fully configured. An example is shown in Figure 17.


The tool configures the server endpoint of a connection within device B. The tool sends an Apply Request, which causes the transition to the Established rate and activates the Inactivity/Watchdog Timer.

The tool configures the client endpoint of the connection. While performing this configuration, the Server endpoint times out due to the fact that a low expected_packet_rate was specified.

Figure 17 – Using tools for configuration

This rule takes into consideration that the application-to-application connection is not really established until the associated information exchange is performed for the first time.

2. All subsequent activations of the Inactivity/Watchdog Timer use the **expected_packet_rate** multiplied by the **connection_timeout_multiplier** as the number of milliseconds to load into the Inactivity/Watchdog Timer.

NOTE 2 If the **expected_packet_rate** attribute specifies the value zero (0), then the Inactivity/Watchdog Timer is not activated and/or used by the Connection object. A Connection object whose **expected_packet_rate** attribute is zero (0) will never experience an Inactivity/Watchdog timeout.

Production inhibit timer

This timer is required to be managed by the Connection object within the **Client end-point** of an I/O Connection when the **production_inhibit_time** attribute value is non-zero. This timer is started when data is produced by the Connection object. The Connection object shall not produce new data if this timer is running. A retry may, however, be sent to the Link Producer. Expiration of this timer allows the Connection object to send new data.

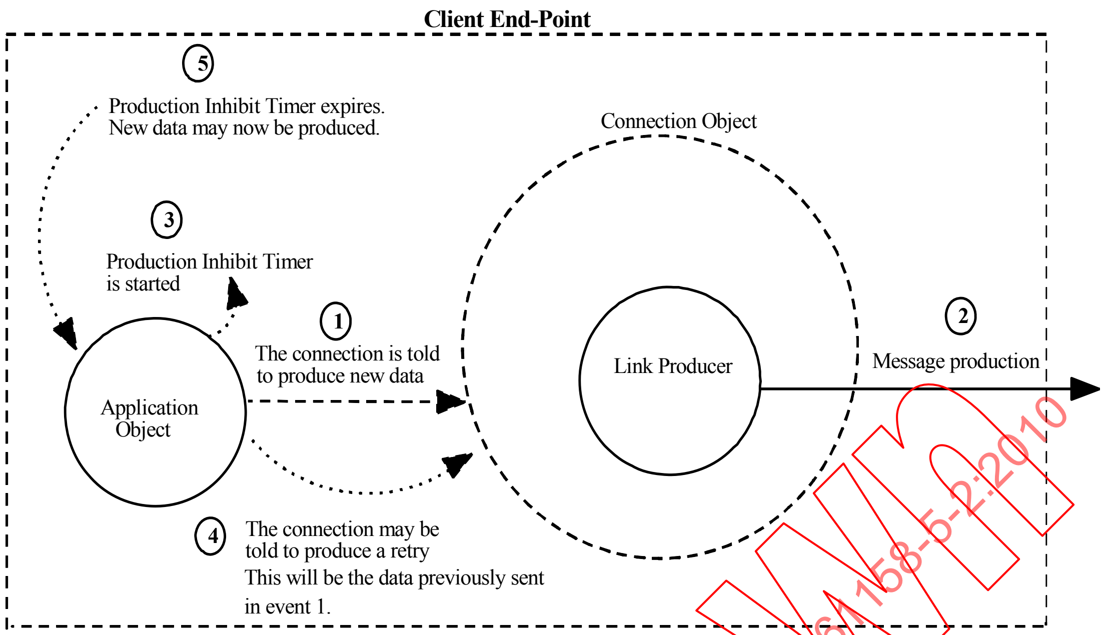


Figure 18 – Production inhibit timer

Important: The Production Inhibit Timer (see Figure 18) is initialized with the value in the **production_inhibit_time** attribute. If the **production_inhibit_time** attribute specifies the value zero (0), then the Production Inhibit Timer is not activated and/or used by the Client end-point.

6.2.3.3 Connection ASE service specification

6.2.3.3.1 Supported services

This subclause contains the definition of services that are unique to this ASE. The services defined for this ASE are

Connection_Bind
Producing_Application_Lookup
SafetyOpen
SafetyClose.

6.2.3.3.2 Connection service common parameters

The FAL service common parameters specified in 6.2.1.3.2 are also used with Connection services.

6.2.3.3.3 Connection service status codes

Specific status codes for the Connection are detailed in IEC 61158-6-2:2010.

6.2.3.3.4 Connection_Bind

6.2.3.3.4.1 Service overview

The Connection_Bind object specific service binds two dynamically created I/O connection instances together for purposes of connection timeouts and deletions.

6.2.3.3.4.2 Service primitives

The service parameters for this service are shown in Table 48.

Table 48 – Connection_Bind service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Bound Instances	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Bound Instances

Instance numbers of Connection objects to be bound.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 18).

6.2.3.3.4.3 Service procedure

No specific service procedure.

6.2.3.3.5 Producing_Application_Lookup

6.2.3.3.5.1 Service overview

The Producing_Application_Lookup object specific service provides a mechanism to find one or more connection instances in the Established state producing data from a given application object.

6.2.3.3.5.2 Service primitives

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

Table 49 – Service_Name service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Local	S			
UCMM Record identifier	S			
Transport identifier	S			
Receiver/Server Local ID		M		
Path	M	C		
Routing Path	M			
Additional Path	U	U(=)		
Port ID		M		
Producing Application Path	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Instance Count			M	M(=)
Connection Instance List			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Producing application path

Connection path of producing application to be searched for within connections in the Established state.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter indicates success.

Instance count

Number of Instances returned in the Connection Instance List parameter.

Connection instance list

List of instance numbers of connection producing the data from the requested connection path within the node.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter indicates an error (see Table 18)

6.2.3.3.5.3 Service procedure

No specific service procedure.

6.2.3.3.6 SafetyOpen

See IEC 61784-3-2 for the definition of this service.

6.2.3.3.7 SafetyClose

See IEC 61784-3-2 for the definition of this service.

6.3 ARs

6.3.1 Overview

6.3.1.1 General

One of the roles of the application layer is to establish and maintain connections. A connection is similar to a telephone circuit. When a call is placed, the telephone system selects a path for your call and sets up each switching station in the route to handle it. As long as the call continues, the resulting virtual circuit remains open, carrying data or voice traffic. In the telephone system, a single call can travel over multiple, different type links, and the format of the data can change from link to link, but the connection appears the same to both the caller and the party called. Sound in one end becomes sound out the other.

A connected message assumes previously negotiated resources and parameters at its source, its destination(s), and any intermediate transit points. These resources are referenced by a unique connection identifier and do not need to be contained in each message, only the connection ID is needed to identify the message and refer to its related parameters, thus giving significant savings in message efficiency.

An unconnected message provides a means to communicate on the local link without previously negotiated resources at the destination so it shall carry full destination ID details, internal data descriptors and full source ID details if a reply is requested. Unconnected messages are used mainly to create connections.

The unconnected service is provided by the Unconnected Message Manager (UCMM). Messages received through the UCMM are forwarded to the Message Router (MR), which direct them to the appropriate internal object for execution. Connections are established by specific unconnected messages sent through the Connection Manager (CM) using UCMM services. Connections may be established either to the Message Router (for messaging purpose), or directly to an application object. Connection target is specified using a connection path, and may be either on the local or a remote link, through several intermediate router nodes. Once a connection is established with an application object, the UCMM, MR and

CM are no longer required, since data will be exchanged directly with the connected objet, based on the corresponding connection ID. Connected messages sent to the Message Router will be forwarded to the appropriate internal object for execution.

The connected and unconnected protocols specified within this part of IEC 61158 are collectively known as the common industrial protocol (CIP).

6.3.1.2 Message Router

The Message Router object provides a messaging interface through which a client may address a service to any object class or instance residing in the device. Every node shall contain instance 1 of the Message Router object.

6.3.1.3 Application connections

6.3.1.3.1 Connections

An application connection is a logical connection between two applications, and is made at the application layer of the communication model. An application connection is built on one or two transport connections. The originating application is responsible for creating and binding the client transport instance to the originating application, and the target application is responsible for creating and binding the server transport instance to the target application. An application connection includes the transport connection which, in turn, includes the producers and consumers involved in the network connection.

Since the unidirectional class 0 and 1 transports require a single network connection, two unidirectional transports can be established within a single connection. The two unidirectional transports do not have to be coordinated at either of the applications, but shall be linked in any of the routers.

The application connection can also include additional parameters or data. A target application, for example, may need information on electronic keying, ownership verification, or the detailed configuration to establish the connection. Such information is sent as a structure in a data segment as part of the connection path. The data segment is forwarded by the intermediate nodes and delivered to the target application as additional segments in the CM_open indication service primitive. The data segment shall be interpreted by the target application.

NOTE The target application may check the module type in the electronic key, check or save the ownership information, and use the configuration information to set up the device. Success and reply information or failure and a reason for the failure can be returned in the message reply. The format of the data segment is a function of the target application.

6.3.1.3.2 Production trigger, transport class and CM_RPI

A client application uses the transport class, trigger type and CM_RPI to determine when to sample and send data.

The system supports three types of triggers for the production of data: cyclic, change of state, and application–triggered. These triggers only apply to the client application.

The triggers are used in conjunction with the transport class and CM_RPI to control when data are sampled and sent on the link. Table 50 shows how these factors relate to each other in determining when data are produced.

Table 50 – How production trigger, transport class, and CM_RPI determine when data is produced

Trigger type	Transport class	When data are sampled and sent	When data are repeated
Cyclic	0 – Null	At the CM_RPI	
	1 – Duplicate Detection	At the CM_RPI	
	2 – Acknowledged	At the CM_RPI	At 1/4 the CM_RPI or faster if not acknowledged
	3 – Verified	At the CM_RPI	At 1/4 the CM_RPI or faster if not verified
	4 – Non-Blocking	Trigger type not supported	Trigger type not supported
	5 – Fragmenting	Trigger type not supported	Trigger type not supported
	6 - Multipoint Fragmenting	Trigger type not supported	Trigger type not supported
Change of State	0 – Null	When data changes	At the CM_RPI
	1 – Duplicate Detection	When data changes	At the CM_RPI
	2 – Acknowledged	When data changes	At 1/4 the CM_RPI or faster until acknowledged, and then at the CM_RPI
	3 – Verified	When data changes	At 1/4 the CM_RPI or faster until verified, and then at the CM_RPI
	4 – Non-Blocking	When application places on queue	At 1/4 the CM_RPI or faster until acknowledged, and then at the CM_RPI
	5 – Fragmenting	When application places on queue	At 1/4 the CM_RPI or faster until acknowledged, and then at the CM_RPI
	6 - Multipoint Fragmenting	When application generates Write event	At 1/4 the CM_RPI or faster until verified, and then at the CM_RPI
Application	0 – Null	Determined by application	
	1 – Duplicate Detection	Determined by application	
	2 – Acknowledged	Determined by application	At 1/4 the CM_RPI or faster if not acknowledged
	3 – Verified	Determined by application	At 1/4 the CM_RPI or faster if not verified
	4 – Non-Blocking	When application places on queue	At 1/4 the CM_RPI or faster if until acknowledged, and then when application generates Trigger event
	5 – Fragmenting	When application places on queue	At 1/4 the CM_RPI or faster if until acknowledged, and then when application generates Trigger event
	6 - Multipoint Fragmenting	When application generates Trigger event	At 1/4 the CM_RPI or faster if until verified, and then when application generates Trigger event

6.3.1.3.3 Polling

Although related to triggering, polling is a server–only function. The system supports two types of polling: asynchronous and synchronous.

Asynchronous polling uses transport class 2 (acknowledged). When the server application receives the poll request, the current data buffer is sent as the acknowledgement. This method requires the application to continuously sample data and update the buffer. The application sampling the data does not know which sample is sent in response to [via] a poll request.

The synchronous poll shall use transport class 3 (verified). When the server application is notified of the poll request, it shall sample new data, which it returns as the verification.

6.3.1.4 Transport connections

6.3.1.4.1 General

A transport connection is a logical binding between two transport instances that is made to enable two applications to transfer data over network connections. A transport instance is a specific implementation of a transport function. Every transport connection is built on one or two network connections. Figure 19 uses the communication model to show the context in which transport connections are made.

NOTE 1 Input, Download, Upload and Scanner functions are examples of users of the transport services, they are not described in IEC 61158 series.

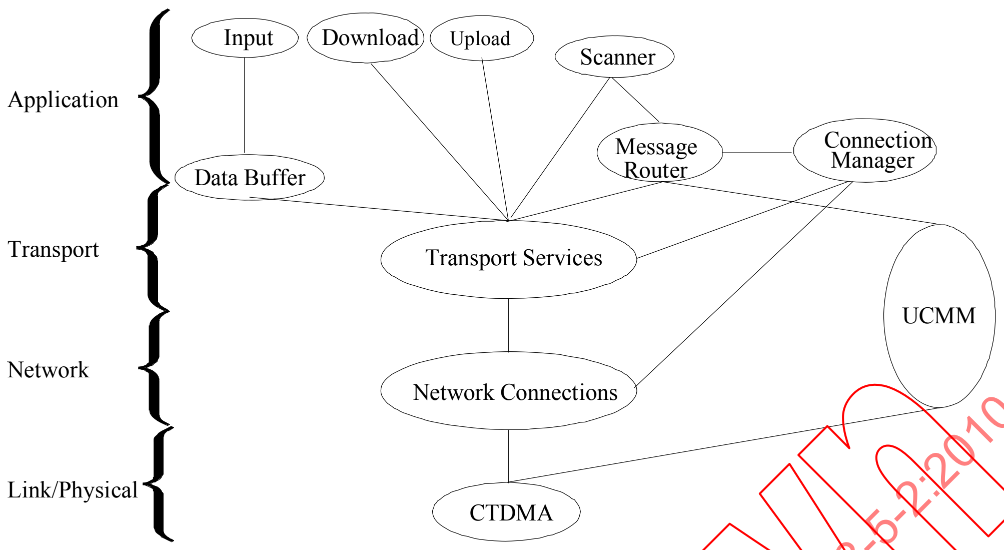


Figure 19 – Context of transport services within the connection model

In Figure 20, application A interfaces to a client transport instance to trigger the production of data over the network connection, and application B interfaces to a server transport instance to be notified that data has been written to the transport protocol data unit (T–PDU) buffer. A T–PDU comprises one packet of data as well as its link, network, and transport headers. A T–PDU buffer is a memory location in which one T–PDU can be stored.

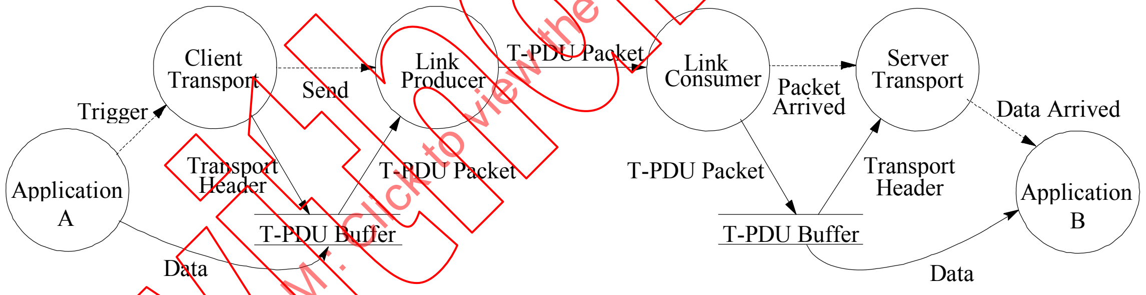


Figure 20 – Application–to–application view of data transfer

Data is not directly written to or read from the transport instances: data is sent to or removed from the T–PDU buffers. The client and server transport instances are responsible only for managing the transport headers of the data packets in the T–PDU buffers. See specification of transport classes in IEC 61158-6-2:2010, 9.3 for additional details on the responsibilities of the client and server transport instances.

NOTE 2 Producers, consumers, and transport instances do not have public interfaces. They are accessible only through the Connection Manager that creates them.

6.3.1.4.2 Components of transport connections

6.3.1.4.2.1 Components

A transport connection includes the transport instance(s), the client– and server–side T–PDU buffer(s), and the network connection(s) involved. The network connection, in turn, includes the link producer(s) and link consumer(s). The originating application is responsible for creating all of the above components, and for creating bindings between those components that result in the desired transport connection being created.