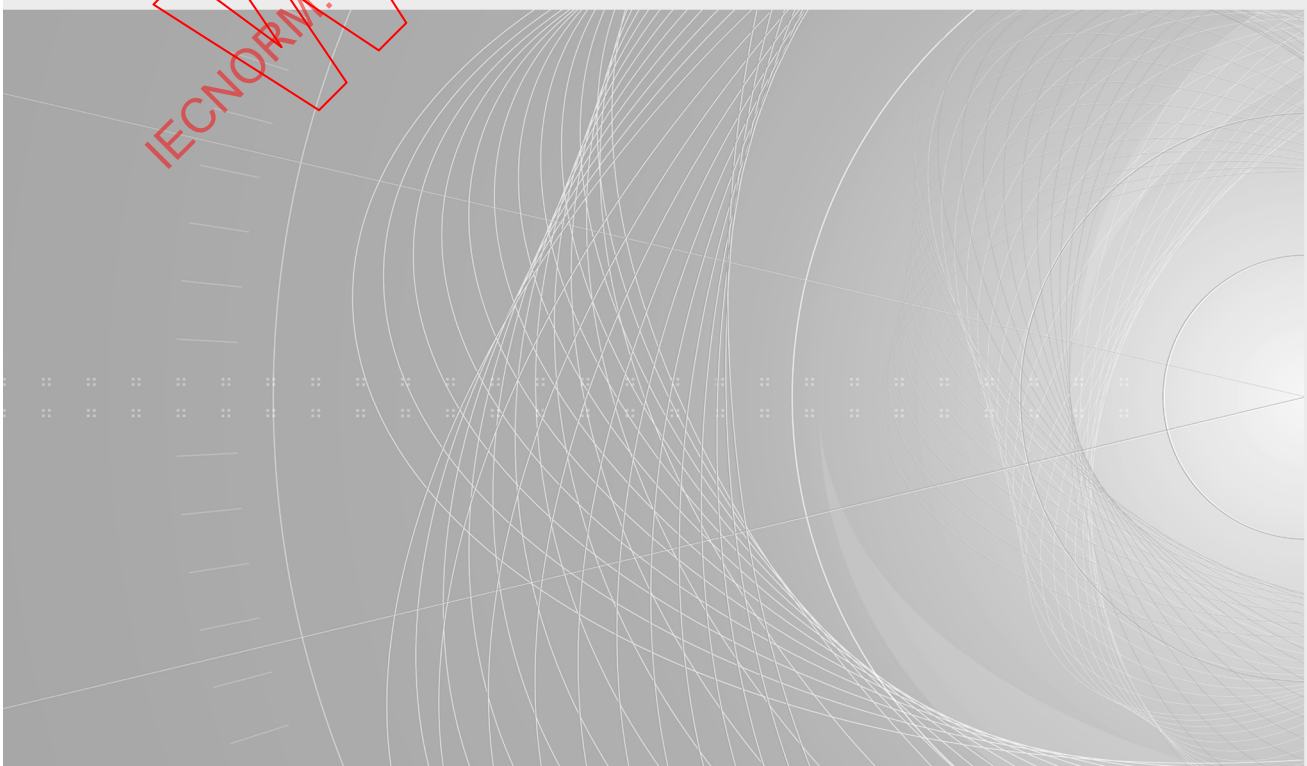
IEC 61158-5-10

Edition 2.0 2010-08

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 5-10: Application layer service definition – Type 10 elements



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

About the IEC

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

About IEC publications

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

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 It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

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

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

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

IEC 61158-5-10

Edition 2.0 2010-08

INTERNATIONAL STANDARD

colour inside

Industrial communication networks – Fieldbus specifications – Part 5-10: Application layer service definition – Type 10 elements

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE XH

ICS 25.04.40; 35.100.70; 35.110

ISBN 978-2-88912-107-6

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-10: Application layer service definition – Type 10 elements

FOREWORD

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

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

International Standard IEC 61158–5–10 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:

- corrections;
- improvements;
- optimization of the synchronization;
- optimization of the startup time from power down.

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 5-10: Application layer service definition – Type 10 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 10 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 10 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 10 IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) 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 do they constrain the implementations of application layer entities within industrial automation systems.

There is conformance of equipment to this application layer service definition standard mainly achieved through implementation of the modeled behavior of an application layer user (e.g. see user state machines) accompanied by implementation of conforming application layer protocols that fulfill the 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 61131-1, *Programmable controllers – Part 1: General information*

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

IEC 61158-5-3:2010[1], *Industrial communication networks – Fieldbus specifications – Part 5-3: Application layer service definitions – Type 3 elements*

IEC 61158-6-3:2010[1], *Industrial communication networks – Fieldbus specifications – Part 6-3: Application layer protocol specification – Type 3 elements*

IEC 61158-6-10:2010[1], *Industrial communication networks – Fieldbus specifications – Part 6-10: Application layer protocol specification – Type 10 elements*

IEC 61375-1, *Electric railway equipment – Train bus – Part 1:Train communication network*

---

1 To be published.

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 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

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

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

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

ISO/IEC 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 10646-1, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

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

IEEE 802-2001, *IEEE Standard for Local and metropolitan area networks: Overview and architecture,* available at <http://www.ieee.org>

IEEE 802.1AB-2005, *IEEE Standard for Local and metropolitan area networks: Station and media access control connectivity discovery,* available at <http://www.ieee.org>

IEEE 802.1D-2004, *IEEE Standard for Local and Metropolitan Area Networks – Media Access Control (MAC) Bridges,* available at <http://www.ieee.org>

IEEE 802.1Q-2005, *IEEE Standard for Local and metropolitan area networks – Virtual bridged local area networks,* available at <http://www.ieee.org>

IEEE 802.3-2005, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer,* available at <http://www.ieee.org>

IETF RFC 768, *User Datagram Protocol*, available at <http://www.ietf.org>

IETF RFC 791, *Internet Protocol*, available at <http://www.ietf.org>

IETF RFC 792, *Internet Control Message Protocol*, available at <http://www.ietf.org>

IETF RFC 826, *An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*, available at <http://www.ietf.org>

IETF RFC 1034, *Domain names – concepts and facilities*, available at <http://www.ietf.org>

IETF RFC 1112, *Host Extensions for IP Multicasting*, available at <http://www.ietf.org>

IETF RFC 1305, *Network Time Protocol (Version 3) – Specification, Implementation and Analysis*, available at <http://www.ietf.org>

IETF RFC 1952, *GZIP file format specification version 4.3*, available at <http://www.ietf.org>

IETF RFC 2131, *Dynamic Host Configuration Protocol*; available at <http://www.ietf.org>

IETF RFC 2365, *Administratively Scoped IP Multicast*, available at <http://www.ietf.org>

IETF RFC 2674, *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions*, available at <http://www.ietf.org>

IETF RFC 2737, *Entity MIB (Version 2)*, available at <http://www.ietf.org>

IETF RFC 2863, *The Interfaces Group MIB*, available at <http://www.ietf.org>

IETF RFC 3330, *Special-Use IPv4 Addresses*, available at <http://www.ietf.org>

IETF RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*, available at <http://www.ietf.org>

IETF RFC 3490, *Internationalizing Domain Names in Applications (IDNA)*, available at <http://www.ietf.org>

IETF RFC 3621, *Power Ethernet MIB*, available at <http://www.ietf.org>

IETF RFC 3636, *Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)*, available at <http://www.ietf.org>

The Open Group – Publication C706, *Technical standard DCE1.1: Remote Procedure Call*, available at <http://www.opengroup.org/onlinepubs/9629399/toc.htm>

3 Terms, definitions, abbreviations, symbols and conventions

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

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

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

3.1.2 ISO/IEC 8822 terms

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

a) abstract syntax
b) presentation context

3.1.3 ISO/IEC 9545 terms

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

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

3.1.4 ISO/IEC 8824-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824-1 apply:

a) object identifier
b) type

3.2 Additional terms and definitions for distributed automation

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

3.2.1
active connection control object (ACCO)
instance of a certain FAL class that abstracts the interconnection facility (as Consumer and Provider) of an automation device

3.2.2
configuration data base
interconnection information maintained by the ACCO ASE

3.2.3
connection
the logical link between sink and source of attributes and services at different custom interfaces of Custom RT-Auto objects

3.2.4
connection channel
description of a connection between a sink and a source of data items

3.2.5
consumer
node or sink that is receiving data from a producer

3.2.6
consumerID
unambiguous identifier within the scope of the ACCO assigned by the consumer to recognize the internal data of a configured interconnection sink.

3.2.7
data marshaling
the encoding of parameters of the FAL service primitives with respect to their interface definition

NOTE This is part of the abstract ORPC model.

3.2.8
engineering
abstract term that characterizes the client application or device responsible for configuring an automation system via interconnecting data items

3.2.9
event
instance of a change of conditions

3.2.10
interface
collection of FAL class attributes and services that represents a specific view on the FAL class

3.2.11
interface definition language
syntax and semantic of describing service parameters in a formal way

NOTE This description is the input for the ORPC model, especially for the ORPC wire protocol.

3.2.12
interface pointer
key attribute that unambiguously addresses an object interface instance

3.2.13
logical device
certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.2.14
method
<object> a synonym for an operational service which is provided by the server ASE and invoked by a client

3.2.15
object remote procedure call
model for object oriented or component based remote method invocation

3.2.16
physical device
certain FAL class that abstracts the hardware facilities of an automation device

3.2.17
property
synonym for ASE attributes which are readable or writeable via operational ASE services

NOTE These services are generally named “get_<Attribute Name>" or “set_<Attribute Name and correspond with the IDL keywords “propget and “propput.

3.2.18
provider
source of a data connection.

3.2.19
providerID
an unambiguous identifier within the scope of the ACCO assigned by the provider to recognize the internal data of a configured connection source

3.2.20
quality code
additional status information of a data item

3.2.21
quality code aware
attribute of the RT-Auto class that indicates that an RT-Auto object uses a status code for its data items

3.2.22
quality code unaware
opposite of quality code aware

3.2.23
RT-Auto
FAL class that abstracts the automation function as a process-related component of an automation device

3.2.24
runtime object model
objects that exist in a device together with their interfaces and methods that are accessible

3.3 Additional terms and definitions for decentralized periphery

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

3.3.1
alarm
activation of an event that shows a critical state

3.3.2
alarm ack
acknowledgment of an event that shows a critical state

3.3.3
alarm data object
object(s) which represent critical states referenced by device/slot/subslot/alarm type

3.3.4
allocate
take a resource from a common area and assign that resource for the exclusive use of a specific entity

3.3.5
application
function or data structure for which data is consumed or produced

3.3.6
application layer interoperability
capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.3.7
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.3.8
application process identifier
distinguishes multiple application processes used in a device

NOTE Application process identifier is assigned by PROFIBUS International (PI).

3.3.9
application process object
component of an application process that is identifiable and accessible through an FAL application relationship

NOTE Application process object definitions are composed of a set of values for the attributes of their class (see the definition for Application Process Object Class Definition). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions, and to dynamically create and delete application process objects and their corresponding definitions.

3.3.10
application process object class
class of application process objects defined in terms of the set of their network-accessible attributes and services

3.3.11
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

3.3.12
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.3.13
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.

3.3.14
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.

3.3.15
backup
status of the IO AR, which indicates that it, is in the standby state

3.3.16
behavior
indication of how an object responds to particular events

3.3.17
channel
representation of a single physical or logical link of an input or output application process object of a server to the process in order to support addressing of diagnosis information

NOTE The channel typically represents a single connector or clamp as a real interface of a module or sub-module. This reference is used to identify points of failure within diagnosis PDUs.

3.3.18
channel related diagnosis
information concerning a specific element of an input or output application process object, provided for maintenance purposes

EXAMPLE open loop

3.3.19
class
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.

3.3.20
class attributes
attribute that is shared by all objects within the same class

3.3.21
class code
unique identifier assigned to each object class

3.3.22
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.

3.3.23
clear
status of the IO controller, which indicates that the control algorithm is currently not running

3.3.24
client

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

b) initiator of a PDU to which a server reacts

3.3.25
common profile
collection of device independent information and functionality providing consistency between all devices

3.3.26
communication data object
object(s) which are parameter of communication relationships and referenced by device/ slot/ subslot/ index

3.3.27
configuration check
comparison of the expected IO-Data object structuring of the client with the real IO-Data object structuring to the server in the start-up phase

3.3.28
configuration fault
unacceptable difference between the expected IO-Data object structuring and the real IO-Data object structuring, as detected by the server

3.3.29
configuration identifier
representation of a portion of IO Data of a single input- and/or output-module of a server

3.3.30
consume
act of receiving data from a provider

3.3.31
consumer
node or sink receiving data from a provider

3.3.32
context management
network-accessible information (communication objects) that supports managing the operation of the fieldbus system, including the application layer

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

3.3.33
conveyance path
unidirectional flow of APDUs across an application relationship

3.3.34
cyclic
repetitive in a regular manner

3.3.35
data consistency
means for coherent transmission and access of the input- or output-data object between and within client and server

3.3.36
device
physical hardware connected to the link

NOTE A device may contain more than one node.

3.3.37
device ID
vendor assigned device type identification

3.3.38
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.3.39
diagnosis data object
object(s) which contain diagnosis information referenced by device/slot/subslot/index

3.3.40
diagnosis information
all data available at the server for maintenance purposes

3.3.41
dynamic reconfiguration
change of IO data objects without interruption of an established application relationship and continuous updating of non-changed IO data objects

3.3.42
endpoint
one of the communicating entities involved in a connection

3.3.43
engineering
abstract term that characterizes the client application or device responsible for configuring an automation system

3.3.44
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.3.45
error class
general grouping for related error definitions and corresponding error codes

3.3.46
error code
identification of a specific type of error within an error class

3.3.47
event
instance of a change of conditions

3.3.48
extended channel related diagnosis
information concerning a specific element of a specific application process object, provided for maintenance purposes

EXAMPLE Link Fail

3.3.49
fast startup
accelerated procedure to setup an IO device after a power cycle

3.3.50
frame
unit of data transfer on a link

3.3.51
identification data object
object(s) that contain information about device, module and sub-module manufacturer and type referenced by device/slot/subslot/index

3.3.52
implicit AR endpoint
AR endpoint that is defined locally within a device without use of the create service

3.3.53
index
address of a record data object within an application process

3.3.54
instance
the actual physical occurrence of an object within a class that identifies one of many objects within the same object class

3.3.55
instance attributes
attribute that is unique to an object instance and not shared by the object class

3.3.56
instantiated
object that has been created in a device

3.3.57
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.

3.3.58
IO controller
controlling device, which acts as client for several IO devices (field devices)

NOTE This is usually a programmable controller or a distributed control system.

3.3.59
IO data object
object designated to be transferred cyclically for the purpose of processing and referenced by device/slot/subslot

3.3.60
IO device
field device which acts as server for IO operation

3.3.61
IO parameter server
server for application parameter of IO devices (client)

NOTE This is usually a device to backup parameter data and to log online changes of device parameter.

3.3.62
IO subsystem
subsystem composed of one IO controller and all its associated IO devices

3.3.63
IO supervisor
engineering device which manages commissioning and diagnosis of an IO system

3.3.64
IO system
system composed of all its IO subsystems

NOTE As an example a PLC with more than one IO controller (network interface) controls one IO system composed of an IO subsystem for each IO controller.

3.3.65
Isochronous mode
IO system operating tightly synchronized with a jitter of less than 1 µs

3.3.66
member
piece of an attribute that is structured as an element of an array

3.3.67
method
a synonym for an operational service which is provided by the server ASE and invoked by a client

3.3.68
module

hardware or logical component of a physical device

3.3.69
network
set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.3.70
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 a clearly defined interface and behavior

3.3.71
object specific service
service unique to the object class which defines it

3.3.72
operate
status of the IO controller that indicates that the control algorithm is currently running

3.3.73
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.3.74
physical device
automation or other network device

3.3.75
point-to-point connection
connection that exists between exactly two application process objects

3.3.76
primary
status of the IO AR that indicates that it is in the operating state

NOTE Besides a primary IO AR a backup IO AR may exist. In example used for redundancy and dynamic reconfiguration of IO data.

3.3.77
provider
node or source sending data to one or many consumer

3.3.78
PTCP domain
certain number of PTCP subdomains in one IP subnet

3.3.79
PTCP subdomain
certain amount of DTEs with synchronized clocks

3.3.80
record data object
object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing and referenced by device/slot/subslot/index

3.3.81
resource
processing or information capability

3.3.82
run
status of the IO controller which indicates that the control algorithm is currently operating

3.3.83
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

3.3.84
service
operation or function that an object and/or object class performs upon a request from another object and/or object class

3.3.85
slot
address of a structural unit within an IO device

NOTE Within a modular device, a slot typically addresses a physical module. Within compact devices, a slot typically addresses a logical function or virtual module.

3.3.86
stop
status of the IO controller which indicates that the control algorithm is currently not running

3.3.87
submodule
hardware or logical component of a module

3.3.88
subslot
address of a structural unit within a slot

NOTE A subslot may address a physical interface for submodules within a module. Generally, a subslot is a second level to structure data within a device.

3.3.89
vendor ID
central administrative number used as manufacturer identification

NOTE The vendor ID is assigned by PROFIBUS International (PI).

3.4 Additional terms and definitions for media redundancy

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

3.4.1
failure
termination of the ability of an item to perform a required function

NOTE 1 After failure the item has a fault.

NOTE 2 "Failure" is an event, as distinguished from "fault", which is a state.

NOTE 3 This concept as defined does not apply to items considering of software only.

3.4.2
fault
state of an item characterized by its inability to perform a required function, excluding the inability during preventive maintenance or other planned actions, or due to lack of external resources

NOTE A fault is often the result of a failure of the item itself, but may exist without prior failure.

3.4.3
recovery
event when the item regains the ability to perform a required function, after a fault

3.4.4
recovery time
time period required for recovery

3.4.5
redundancy
existence in an item of two or more means of performing a required function

NOTE In this context, the existence of more than one path (consisting of links and switches) between end nodes.

3.4.6
ring
network where each node is connected in series to two other nodes

NOTE 1 Nodes are connected to one another in the logical shape of a circle.

NOTE 2 Frames are passed sequentially between active nodes, each able to examine or modify the frame before forwarding it.

3.4.7
ring port
port of a switch to which a ring link is attached

3.4.8
ring link
link that connects two switches of a ring

3.4.9
switch node
MAC bridge as defined in IEEE 802.1D

NOTE It will be called “switch” in this context.

3.5 Abbreviations and symbols

ACCO	Active Connection Control Object
AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application Process Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
AREP	Application Relationship End Point
ASCII	American Standard code for Information Interchange
ASE	Application Service Element
CID	Connection ID
CIM	Computer Integrated Manufacturing
CIP	Control and Information Protocol
CM_API	Actual Packet Interval
CM_RPI	Requested Packet Interval
Cnf	Confirmation
COR	Connection originator
CR	Communication Relationship
CREP	Communication Relationship End Point
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLL	Data Link Layer
DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	DL-service-data-unit
DNS	Domain Name Service
DP	Decentralised Peripherals
FAL	Fieldbus Application Layer
FIFO	First In First Out
HMI	Human-machine Interface
ID	Identifier
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
Ind	Indication
IP	Internet Protocol
ISO	International Organization for Standardization
LDev	Logical device
LME	Layer Management Entity
O2T	Originator to target (connection characteristics)
OpT	Originator to target (connection characteristics)
ORPC	Object Remote Procedure Call
OSI	Open Systems Interconnect
PDev	Physical device
PDU	Protocol Data Unit
PL	Physical Layer
QoS	Quality of Service
QC	Quality code
REP	Route Endpoint
Req	Request
Rsp	Response
RT	Runtime

SAP	Service Access Point
SCL	Security Level
SDU	Service Data Unit
SEM	State event matrix
SMIB	System Management Information Base
SMK	System Management Kernel
STD	State transition diagram, used to describe object behaviour
S-VFD	Simple Virtual Field Device
T2O	Target to originator (connection characteristics)
TÞO	Target to originator (connection characteristics)
VAO	Variable Object

3.6 Additional abbreviations and symbols for distributed automation

ACCO	Active Connection Control Object
IDL	Interface Definition Language
IP	Internet Protocol
LDev	Logical device
ORPC	Object Remote Procedure Call
PDev	Physical device
QoS	Quality of Service
QC	Quality code
RT	Runtime
TCP	Transmission Control Protocol

3.7 Additional abbreviations and symbols for decentralized periphery

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
ALPMI	Alarm Protocol machine Initiator
ALPMR	Alarm Protocol machine Responder
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
APO	Application Process Object
AR	Application Relationship
AREP	Application Relationship End Point
ARP	Address Resolution Protocol
ASCII	American Standard code for Information Interchange
ASE	Application Service Element
BMC	Best Master Clock
CM	Context Management
Cnf	Confirmation
CR	Communication Relationship
CREP	Communication Relationship End Point
DCE	OSF Distributed Computing Environment
DCP	Discovery and basic Configuration Protocol
DCPMCR	DCP Multicast Receiver
DCPMCS	DCP Multicast Sender
DCPUCR	DCP Unicast Receiver
DCPUCS	DCP Unicast Sender
DHCP	Dynamic Host Configuration Protocol
DIM	Device Interface Module
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLL	Data Link Layer
DLPDU	Data Link-Protocol Data Unit
DLSDU	DL-service-data-unit
DNS	Domain Name Service
DTE	Date Terminal Equipment

FAL	Fieldbus Application Layer
FIFO	First In First Out
GSDML	General Station Description Markup Language
I&M	Identification and Maintenance Profile
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ID	Identifier
IEC	International Electrotechnical Commission
Ind	Indication
IOCS	Input Output Object Consumer Status
IOPS	Input Output Object Provider Status
IP	Internet Protocol
IR	Isochronous Relay
IRT	Isochronous Real Time Protocol
ISO	International Organization for Standardization
IsoM	Isochronous Mode
LED	Light Emitting Diode
LLDP	Link Layer Discovery Protocol
LME	Layer Management Entity
lsb	Least Significant Bit
LT	Length/type
MAC	Medium Access Control
msb	Most Significant Bit
NCA	Network Computing Architecture
OSI	Open Systems Interconnect
PDU	Protocol Data Unit
PI	PROFIBUS International see www.profibus.com
PL	Physical Layer
PTCP	Precision Transparent Clock Protocol
QoS	Quality of Service
Req	Request
RPC	Remote Procedure Call
Rsp	Response
RT	Real Time Protocol
RTA	Real Time Protocol Acyclic
RTC	Real Time Protocol Cyclic
RTE	Real Time Ethernet
SDU	Service Data Unit
TLV	Type Length Value (coding rule)
UDP	User Datagram Protocol
UUID	Universal Unique Identifier
VLAN	Virtual Local Area Network

3.8 Additional abbreviations and symbols for media redundancy

MRC	Media Redundancy Client
MRM	Media Redundancy Manager
MRP	Media Redundancy Protocol

3.9 Conventions

3.9.1 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.

3.9.2 General conventions

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

3.9.3 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 255 are reserved by this standard to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2 048 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 a 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.

3.9.4 Conventions for service definitions

3.9.4.1 General

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

3.9.4.2 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 use 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.

3.9.4.3 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 Concepts

NOTE Clause 9 of IEC/TR 61158–1:2010 describes the concepts of the application layer service descriptions and the templates used in this standard.

5 Data type ASE

5.1 General

5.1.1 Overview

Fieldbus data types specify the machine independent syntax for application data conveyed by FAL services. The fieldbus application layer supports the definition and transfer of both basic and constructed data types. Encoding rules for the data types specified in this clause are provided in IEC 61158–6–10.

Basic types are atomic types that cannot be decomposed into more elemental types. Constructed types are types composed of basic types and other constructed types. Their complexity and depth of nesting is not constrained by this standard.

Data types are defined as instances of the data type class, as shown in Figure 1. Only a subset of the data types defined in this Clause are shown in this figure. Defining new types is accomplished by providing a numeric id and supplying values for the attributes defined for the data type class.

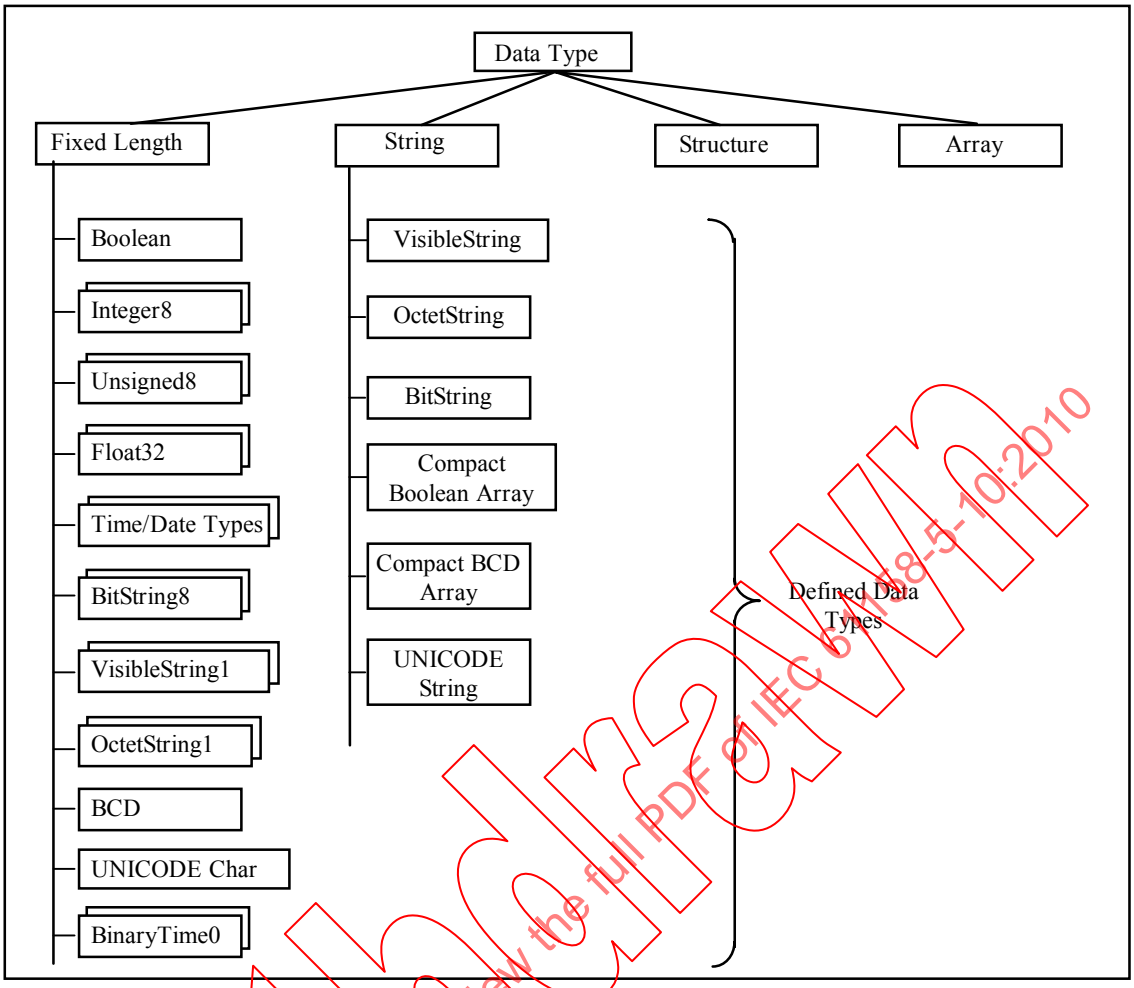


Figure 1 – Data type class hierarchy example

The data type definitions in Figure 1 are represented as a class/format/instance structure beginning with data type class entitled "Data type". The formats for data types are defined by the data type class and are represented in Figure 1.

The basic data classes are used to define fixed length and bitstring data types. Standard types taken from ISO/IEC 8824-1 are referred to as *simple* data types. Other standard basic data types are defined specifically for fieldbus applications and are referred to as *specific types*.

The constructed types specified in this standard are strings, arrays and structures. There are no standard types defined for arrays and structures.

5.1.2 Basic type overview

Most basic types are defined from a set of ISO/IEC 8824–1 types (simple types). Some ISO/IEC 8824–1 types have been extended for fieldbus specific use (specific types).

Simple types are ISO/IEC 8824–1 universal types. They are defined in this standard to provide them with fieldbus class identifiers.

Specific types are basic types defined specifically for use in the fieldbus environment. They are defined as simple class subtypes.

Basic types have a constant length. Two variations are defined, one for defining data types whose length is an integral number of octets, and one for defining data types whose length is bits.

NOTE Boolean, Integer, OctetString, VisibleString, and UniversalTime are defined in this standard for the purpose of assigning fieldbus class identifiers to them. This standard does not change their definitions as specified in ISO/IEC 8824–1.

5.1.3 Fixed length type overview

The length of Fixed length types is an integral number of octets.

5.1.4 Constructed type overview

5.1.4.1 Strings

A string is composed of an ordered set, variable in number, of homogeneously typed fixed-length elements.

5.1.4.2 Arrays

An array is composed of an ordered set of homogeneously typed elements. This standard places no restriction on the data type of array elements, but it does require that each element be of the same type. Once defined, the number of elements in an array may not be changed.

5.1.4.3 Structures

A structure is made of an ordered set of heterogenously typed elements called fields. Like arrays, this standard does not restrict the data type of fields. However, the fields within a structure do not have to be of the same type.

5.1.4.4 Nesting level

This standard permits arrays and structures to contain arrays and structures. It places no restriction on the number of nesting levels allowed. However, the FAL services defined to access data provide for partial access to the level negotiated by the Initiate service. The default number of levels for partial access is one.

When an array or structure contains constructed elements, access to a single element in its entirety is always provided. Access to subelements of the constructed element is also provided, but only when explicitly negotiated during AR establishment or when explicitly preconfigured on pre-established ARs.

NOTE For example, suppose that a data type named "employee" is defined to contain the structure "employee name", and "employee name" is defined to contain "last name" and "first name". To access the "employee" structure, the FAL permits independent access to the entire structure and to the first level field "employee name". Without explicitly negotiating partial access to more than one level, independent access to "last name" or "first name" would not be possible; their values could only be accessed together as a unit through access to "employee" or "employee name".

5.1.5 Specification of user defined data types

Users may find it necessary to define custom data types for their own applications. User defined types are supported by this standard as instances of data type classes.

User defined types are specified in the same manner that all FAL objects are specified. They are defined by providing values for the attributes specified for their class.

5.1.6 Transfer of user data

User data is transferred between applications by the FAL protocol. All encoding and decoding are performed by the FAL user.

The rules for encoding user data in FAL protocol data units is data type dependent. These rules are defined in IEC 61158-6-10. User-defined data types for which there are no encoding rules are transferred as a variable-length sequence of octets. The format of the data within the octet string is defined by the user.

5.2 Formal definition of data type objects

5.2.1 Data type class

5.2.1.1 Template

The data type class specifies the root of the data type class tree. Its parent class "top" indicates the top of the FAL class tree.

FAL ASE:		**DATA TYPE ASE**	
CLASS:		**DATA TYPE**	
CLASS ID:		**5 (FIXED LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)**	
PARENT CLASS:		**TOP**	
ATTRIBUTES:			
1	(m)	Key Attribute:	Data type Numeric Identifier
2	(o)	Key Attribute:	Data type Name
3	(m)	Attribute:	Format (FIXED LENGTH, STRING, STRUCTURE, ARRAY)
4	(c)	Constraint:	Format = FIXED LENGTH | STRING
4.1	(m)	Attribute:	Octet Length
5	(c)	Constraint:	Format = STRUCTURE
5.1	(m)	Attribute:	Number of Fields
5.2	(m)	Attribute:	List of Fields
5.2.1	(o)	Attribute:	Field Name
5.2.2	(m)	Attribute:	Field Data type
6	(c)	Constraint:	Format = ARRAY
6.1	(m)	Attribute:	Number of Array Elements
6.2	(m)	Attribute:	Array Element Data type

5.2.1.2 Attributes

Data type Numeric Identifier
This attribute identifies the numeric identifier of the related data type. The data type numeric identifier has a data type of Unsigned32.

Data type Name
This optional attribute identifies the name of the related data type.

Format
This attribute identifies the data type as a fixed-length, string, array, or data structure.

Octet Length
This conditional attribute defines the representation of the dimensions of the associated type object. It is present when the value of the format attribute is "FIXED LENGTH" or "STRING". For FIXED LENGTH data types, it represents the length in octets. For STRING data types, it represents the length in octets for a single element of a string.

Number of Fields
This conditional attribute defines the number of fields in a structure. It is present when the value of the format attribute is "STRUCTURE".

List of Fields
This conditional attribute is an ordered list of fields contained in the structure. Each field is specified by its number and its type. Fields are numbered sequentially from 0 (zero) in the order in which they occur. Partial access to fields within a structure is supported by identifying the field by number. This attribute is present when the value of the format attribute is "STRUCTURE".

Field Name
This conditional, optional attribute specifies the name of the field. It may be present when the value of the format attribute is "STRUCTURE".

Field Data type
This conditional attribute specifies the data type of the field. It is present when the value of the format attribute is "STRUCTURE". This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric id, or by embedding a constructed data type definition here. When embedding a description, the Embedded Data type description shown below is used.

Number of Array Elements
This conditional attribute defines the number of elements for the array type. Array elements are indexed starting at “0” through “n-1” where the size of the array is “n” elements. This attribute is present when the value of the format attribute is "ARRAY".

Array Element Data type
This conditional attribute specifies the data type for the elements of an array. All elements of the array have the same data type. It is present when the value of the format attribute is "ARRAY". This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric id, or by embedding a constructed data type definition here. When embedding a description, the Embedded Data type description shown below is used.

Embedded Data type Description
This attribute is used to recursively define embedded data types within a structure or array. The template below defines its contents. The attributes shown in the template are defined above in the data type class, except for the Embedded Data type attribute, which is a recursive reference to this attribute. It is used to define nested elements.

ATTRIBUTES:

1	(m)	Attribute:	Format(FIXED LENGTH, STRING, STRUCTURE, ARRAY)
2	(c)	Constraint:	Format = FIXED LENGTH | STRING
2.1	(m)	Attribute:	Data type Numeric ID value
2.2	(m)	Attribute:	Octet Length
3	(c)	Constraint:	Format = STRUCTURE
3.1	(m)	Attribute:	Number of Fields
3.2	(m)	Attribute:	List of Fields
3.2.1	(m)	Attribute:	Embedded Data type Description
4	(c)	Constraint:	Format = ARRAY
4.1	(m)	Attribute:	Number of Array Elements
4.2	(m)	Attribute:	Embedded Data type Description

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 (non zero) and FALSE (zero).

5.3.1.1.2 VARIANT_BOOL

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=2	049
2	Data type Name	=	VARIANT_BOOL
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This data type expresses a Boolean data type with the values TRUE (-1) and FALSE (0) (see Integer16).

5.3.1.2 Bitstring types

5.3.1.2.1 Bit sequence V2

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	115
2	Data type Name	=	V2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This data type expresses a Bitstring data type consisting of 16 single bits combined in two octets and coded according to Table 1.

Each single bit has the values TRUE (1) and FALSE (0).

Table 1 – V2 octets

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

5.3.1.2.2 Nibble L2

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	116
2	Data type Name	=	L2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type expresses a Bitstring data type consisting of four nibbles combined in two octets. A nibble consists of four associated bits. The data type is coded according to Table 2.

The definition of a single nibble is not specified.

Table 2 – L2 octets

Bit	8	7	6	5	4	3	2	1
Octet 1	Nibble 3				Nibble 2			
Octet 2	Nibble 1				Nibble 0			

The definition of a single nibble is not specified.

5.3.1.3 Currency types

NOTE intentionally blank

5.3.1.4 Date types

5.3.1.4.1 date

This data type is the same as Float64.

The data type date has a resolution in the range of one nanosecond. It is valid for dates between 0100-01-01, and 9999-12-31. The value of 0.0 has been defined for 1899-12-30 T 00:00:00. The integer part of the value represents the days after 1899-12-30 (for dates before this day, the corresponding value is negative); the fractional part defines the time at that day.

NOTE Independent of the data type range of date, the devices may support for connections only the data type range of "1900-01-01 T 00:00:00" and greater.

5.3.1.4.2 Date

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	50
2	Data type Name	=	Date
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours with the most significant bit indicating Standard Time or Daylight Saving Time. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned8 data type and gives the year. The values 0 ... 50 correspond to the years 2000 to 2050, the values 51 ... 99 correspond to the years 1951 to 1999.

5.3.1.4.3 TimeOfDay

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	12
2	Data type Name	=	TimeOfDay
3	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.

5.3.1.4.4 TimeOfDay with date indication

This data type is the same as the TimeOfDay data type defined above.

5.3.1.4.5 TimeOfDay without date indication

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	52
2	Data type Name	=	TimeOfDay without date indication
3	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 without date indication. The element is an Unsigned32 data type and gives the time after the midnight in milliseconds.

5.3.1.4.6 TimeDifference

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	13
2	Data type Name	=	TimeDifference
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4 or 6

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The optional second element is an Unsigned16 data type that provides the difference in days.

5.3.1.4.7 TimeDifference with date indication

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	53
2	Data type Name	=	TimeDifference with date indication
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The second element is an Unsigned16 data type that provides the difference in number of days.

5.3.1.4.8 TimeDifference without date indication

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	54
2	Data type Name	=	TimeDifference without date indication
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type is composed of one element of an unsigned value that express the difference in time. The element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds.

5.3.1.5 Enumerated types

5.3.1.5.1 PERSISTDEF

CLASS:	**Data type**		
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2050
2	Data type Name	=	PERSISTDEF
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

The allowed values are shown in Table 3.

Table 3 – PERSISTDEF

Value
CBAVolatile
CBAPendingPersistent
CBAPersistent

5.3.1.5.2 VARTYPE

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2051
2	Data type Name	=	VARTYPE
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

The VARTYPE specifies the data type that governs the interpretation of the data. The allowed values are shown in Table 4.

Table 4 – VARTYPE

Value	Data type
VT_EMPTY	no value
VT_NULL	null value (no valid data)
VT_BOOL	VARIANT_BOOL
VT_I1	char
VT_I2	short
VT_I4	long
VT_UI1	unsigned char
VT_UI2	unsigned short
VT_UI4	unsigned long
VT_R4	float
VT_R8	double
VT_DATE	date
VT_BSTR	BSTR
VT_ERROR	HRESULT
VT_SAFEARRAY_BOOL	SAFEARRAY (VARIANT_BOOL)
VT_SAFEARRAY_I1	SAFEARRAY (char)
VT_SAFEARRAY_I2	SAFEARRAY (short)
VT_SAFEARRAY_I4	SAFEARRAY (long)
VT_SAFEARRAY_UI1	SAFEARRAY (unsigned char)
VT_SAFEARRAY_UI2	SAFEARRAY (unsigned short)
VT_SAFEARRAY_UI4	SAFEARRAY (unsigned long)
VT_SAFEARRAY_R4	SAFEARRAY (float)
VT_SAFEARRAY_R8	SAFEARRAY (double)
VT_SAFEARRAY_DATE	SAFEARRAY (date)
VT_SAFEARRAY_BSTR	SAFEARRAY (BSTR)
VT_SAFEARRAY_ERROR	SAFEARRAY (HRESULT)
VT_USERDEFINED	userdefined struct
VT_DISPATCH	Interface Pointer to an Dispatch interface
VT_UNKNOWN	Interface Pointer to an Unknown interface
NOTE All defined structured data types should use the value VT_USERDEFINED.	

5.3.1.5.3 ITEMQUALITYDEF

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2052
2	Data type Name	=	ITEMQUALITYDEF
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This data type contains the status information of the related data. It consists of three portions: Quality, Substatus and Limits. There are four states of quality (Bad - the value is not useful; Uncertain - the quality of the value is less than normal, but the value may still be useful; Good (Non Cascade) - the quality of the value is good, possible alarm conditions may be indicated by the substatus; Good (Cascade) - the value may be used in control), a set of sub-status values for each quality, and four states of the limits (OK - the value is free to move; low limited (LL) - the value has acceded its low limits, high limited (HL) - the value has acceded its high limits; constant (C) - the value cannot move, no matter what the process does). The allowed values are shown in Table 5.

Table 5 – ITEMQUALITYDEF

Quality	Value	Description
Bad	BadNonSpecific	There is no specific reason why the value is bad. Used for propagation.
	BadNonSpecificLL	and low limited
	BadNonSpecificHL	and high limited
	BadNonSpecificC	and constant
	BadConfigurationError	Set if the value is not useful because there is some other problem with the block, depending on what a specific producer can detect.
	BadConfigurationErrorLL	and low limited
	BadConfigurationErrorHL	and high limited
	BadConfigurationErrorC	and constant
	BadNotConnected	Set if this input is required to be connected and is not connected.
	BadNotConnectedLL	and low limited
	BadNotConnectedHL	and high limited
	BadNotConnectedC	and constant
	BadDeviceFailure	Set if the source of the value is affected by a device failure.
	BadDeviceFailureLL	and low limited
	BadDeviceFailureHL	and high limited
	BadDeviceFailureC	and constant
	BadSensorFailure	Set if the device can determine this condition. The limits define which direction has been exceeded.
	BadSensorFailureLL	and low limited
	BadSensorFailureHL	and high limited
	BadSensorFailureC	and constant
	BadLastKnownValue	Set if this value had been set by communication, which has now failed.
	BadLastKnownValueLL	and low limited
	BadLastKnownValueHL	and high limited
	BadLastKnownValueC	and constant
	BadORPCmFailure	Set if there has never been any communication with this value since it was last Out of Service.
	BadORPCmFailureLL	and low limited
	BadORPCmFailureHL	and high limited
	BadORPCmFailureC	and constant
	BadOutOfService	The value is not reliable because the block is not being evaluated, and may be under construction by a configuration tool. It is set if the block mode is O/S.
	BadOutOfServiceLL	and low limited
	BadOutOfServiceHL	and high limited
	BadOutOfServiceC	and constant
Uncertain	UncertainNonSpecific	There is no specific reason why the value is uncertain. Used for propagation.
	UncertainNonSpecificLL	and low limited
	UncertainNonSpecificHL	and high limited
	UncertainNonSpecificC	and constant
	UncertainLastUsableValue	Whatever was writing this value has stopped doing so. This is used for fail safe handling.

Quality	Value	Description
	UncertainLastUsableValueLL	and low limited
	UncertainLastUsableValueHL	and high limited
	UncertainLastUsableValueC	and constant
	UncertainSubstituteSet	Predefined value is used instead of the calculated one. This is used for fail safe handling.
	UncertainSubstituteSetLL	and low limited
	UncertainSubstituteSetHL	and high limited
	UncertainSubstituteSetC	and constant
	UncertainInitialValue	Value of volatile parameters during and after the reset of the device or a parameter.
	UncertainInitialValueLL	and low limited
	UncertainInitialValueHL	and high limited
	UncertainInitialValueC	and constant
	UncertainSensorNotAccurate	Set if the value is at one of the sensor limits. The limits define which direction has been exceeded. Also set if the device can determine that the sensor has reduced accuracy (e.g. degraded analyzer), in which case no limits are set.
	UncertainSensorNotAccurateLL	and low limited
	UncertainSensorNotAccurateHL	and high limited
	UncertainSensorNotAccurateC	and constant
	UncertainEngineeringUnitsExceeded	Set if the value lies outside of the range of values defined for this parameter. The limits define which direction has been exceeded.
	UncertainEngineeringUnitsExceededLL	and low limited
	UncertainEngineeringUnitsExceededHL	and high limited
	UncertainEngineeringUnitsExceededC	and constant
	UncertainSubNormal	Set if a value derived from multiple values has less than the required number of Good sources.
	UncertainSubNormalLL	and low limited
	UncertainSubNormalHL	and high limited
	UncertainSubNormalC	and constant
	UncertainConfigurationError	Set if there is some inconsistency regarding the parameterization or configuration, depending on what a specific producer can detect.
	UncertainConfigurationErrorLL	and low limited
	UncertainConfigurationErrorHL	and high limited
	UncertainConfigurationErrorC	and constant
	UncertainSimulatedValue	Set when the process value is written by the operator while the block is in manual mode.
	UncertainSimulatedValueLL	and low limited
	UncertainSimulatedValueHL	and high limited
	UncertainSimulatedValueC	and constant
	UncertainSensorCalibration	Set during the active calibration process together with the current measured value.
	UncertainSensorCalibrationLL	and low limited
	UncertainSensorCalibrationHL	and high limited
	UncertainSensorCalibrationC	and constant
Good Non Cascade	GoodNonCascOk	No error or special condition is associated with this value.
	GoodNonCascOkC	and constant
	GoodNonCascActiveUpdateEvent	Set if the value is good and the block has an active Update Event.
	GoodNonCascActiveUpdateEventLL	and low limited
	GoodNonCascActiveUpdateEventHL	and high limited
	GoodNonCascActiveUpdateEventC	and constant
	GoodNonCascActiveAdvisoryAlarm	Set if the value is good and the block has an active Alarm with a priority less than 8.
	GoodNonCascActiveAdvisoryAlarmLL	and low limited
	GoodNonCascActiveAdvisoryAlarmHL	and high limited
	GoodNonCascActiveAdvisoryAlarmC	and constant

Quality	Value	Description
	GoodNonCascActiveCriticalAlarm	Set if the value is good and the block has an active Alarm with a priority greater than or equal to 8.
	GoodNonCascActiveCriticalAlarmLL	and low limited
	GoodNonCascActiveCriticalAlarmHL	and high limited
	GoodNonCascActiveCriticalAlarmC	and constant
	GoodNonCascUnackUpdateEvent	Set if the value is good and the block has an unacknowledged Update Event.
	GoodNonCascUnackUpdateEventLL	and low limited
	GoodNonCascUnackUpdateEventHL	and high limited
	GoodNonCascUnackUpdateEventC	and constant
	GoodNonCascUnackAdvisoryAlarm	Set if the value is good and the block has an unacknowledged Alarm with a priority less than 8.
	GoodNonCascUnackAdvisoryAlarmLL	and low limited
	GoodNonCascUnackAdvisoryAlarmHL	and high limited
	GoodNonCascUnackAdvisoryAlarmC	and constant
	GoodNonCascUnackCriticalAlarm	Set if the value is good and the block has an unacknowledged Alarm with a priority greater than or equal to 8.
	GoodNonCascUnackCriticalAlarmLL	and low limited
	GoodNonCascUnackCriticalAlarmHL	and high limited
	GoodNonCascUnackCriticalAlarmC	and constant
	GoodNonCascInitialFailSafe	This value is from a block that wants its following output block (e.g. AO) to go to Fail Safe.
	GoodNonCascInitialFailSafeLL	and low limited
	GoodNonCascInitialFailSafeHL	and high limited
	GoodNonCascInitialFailSafeC	and constant
	GoodNonCascMaintenanceRequired	The device works still without failure but service support will be necessary soon. This may be detected e.g. by a Transducer Block of a value of pH meter.
	GoodNonCascMaintenanceRequiredLL	and low limited
	GoodNonCascMaintenanceRequiredHL	and high limited
	GoodNonCascMaintenanceRequiredC	and constant
Good Cascade	GoodCascOk	No error or special condition is associated with this value.
	GoodCascOkC	and constant
	GoodCascInitializationAcknowledge	The value is an initialized value from a source (cascade input, remote-cascade in, and remote-output in parameters).
	GoodCascInitializationAcknowledgeLL	and low limited
	GoodCascInitializationAcknowledgeHL	and high limited
	GoodCascInitializationAcknowledgeC	and constant
	GoodCascInitializationRequest	The value is an initialization value for a source (back calculation input parameter), because the lower loop is broken or the mode is wrong.
	GoodCascInitializationRequestLL	and low limited
	GoodCascInitializationRequestHL	and high limited
	GoodCascInitializationRequestC	and constant
	GoodCascNotInvited	The value is from a block which does not have a target mode that would use this input. This covers all cases other than Fail Safe Active, Local Override, and Not Selected. The target mode can be the next permitted mode of higher priority in the case of shedding a supervisory computer.
	GoodCascNotInvitedLL	and low limited
	GoodCascNotInvitedHL	and high limited
	GoodCascNotInvitedC	and constant
	GoodCascDoNotSelect	The value is from a block which shall not be selected, due to conditions in or above the block.
	GoodCascDoNotSelectLL	and low limited
	GoodCascDoNotSelectHL	and high limited
	GoodCascDoNotSelectC	and constant
	GoodCascLocalOverride	The value is from a block that has been locked out by a local key switch or is a Complex AO/DO with interlock logic active. The failure of normal control shall be propagated to a PID

Quality	Value	Description
		block for alarm and display purposes. This also implies Not Invited.
	GoodCascLocalOverrideLL	and low limited
	GoodCascLocalOverrideHL	and high limited
	GoodCascLocalOverrideC	and constant
	GoodCascInitiateFailSafe	The value is from a block that wants its downstream output block (e.g. AO) to go to Fail Safe. This is determined by a block option to initiate Fail Safe if the status of the primary input and/or cascade input goes Bad.
	GoodCascInitiateFailSafeLL	and low limited
	GoodCascInitiateFailSafeHL	and high limited
	GoodCascInitiateFailSafeC	and constant

5.3.1.5.4 STATEDEF

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2053
2	Data type Name	=	STATEDEF
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

The allowed values are shown in Table 6.

Table 6 – STATEDEF

Value
CBANonExistent
CBAInitializing
CBAReady
CBAOperating
CBADefect

5.3.1.5.5 GROUPERRORDEF

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2054
2	Data type Name	=	GROUPERRORDEF
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

The allowed values are shown in Table 7.

Table 7 – GROUPERRORDEF

Value
CBANonAccessible
CBAOkay
CBAProblem
CBAUnknown

5.3.1.5.6 ACCESSRIGHTSDEF

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2055
2	Data type Name	=	ACCESSRIGHTSDEF

3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

The allowed values are shown in Table 8.

Table 8 – ACCESSRIGHTSDEF

Value
CBANoAccess
CBAReadAccess
CBAWriteAccess
CBAFullAccess

5.3.1.6 Handle types

5.3.1.6.1 HRESULT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2056
2	Data type Name	=	HRESULT
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

The allowed values are shown in Table 9. The defined HRESULT values may be extended by user specific values. If necessary the coding scheme as defined in this standard should be applied. In particular, the severity code to indicate success or error shall be used.

Table 9 – HRESULT

Value	Description
CBA_E_ACCESSBLOCKED	Access to this item is blocked because a constant is currently assigned to it.
CBA_E_CAPACITYEXCEEDED	The request exceeds the resources according to the machines capacity.
CBA_E_COUNTEXCEEDED	The request exceeds the resources according to their count.
CBA_E_CRDATALENGTH	The length of the requested data for this CR is invalid for RT.
CBA_E_DEFECT	Hardware defect detected, replacement needed.
CBA_E_DISCONNECTRUNNING	The request exceeds the resources and there is a Disconnect running which will free some resources later.
CBA_E_FLAGUNSUPPORTED	One of the flags used are not supported in this implementation.
CBA_E_FRAMECOUNTUNSUPPORTED	The number of RT frames transferred between two ACCOs at the same QoS is restricted by the implementation.
CBA_E_INUSE	The destination specified in the item identifier is already connected to some other provider.
CBA_E_INVALIDCONNECTION	It is not allowed to specify a connection from an identifier to itself.
CBA_E_INVALIDCOOKIE	The Cookie value is not valid.
CBA_E_INVALIDENUMVALUE	The enumeration value is invalid.
CBA_E_INVALIDEPSILON	The Epsilon type or value is not valid.
CBA_E_INVALIDID	The ConsumerID or ProviderID is not valid.
CBA_E_INVALIDSUBSTITUTE	The Substitute type or value is not valid.
CBA_E_ITEMTOOLARGE	The size of the item exceeds the upper limit for RT.
CBA_E_LIMITVIOLATION	The data exceeds limits of the corresponding data type.
CBA_E_LINKFAILURE	The network cards current link status is unsuitable for this protocol.
CBA_E_LOCATIONCHANGED	The location of the restored connection differs from the location of the saved one.
CBA_E_MALFORMED	The identifier is malformed (too long, unallowed characters, no separation characters, syntactically invalid)
CBA_E_MODECHANGE	The consumer tried to change the data transfer mode from push (OnData-Changed) to pull (GetConnectionData) or vice versa.
CBA_E_NONACCESSIBLE	The item is not accessible.
CBA_E_NOTAPPLICABLE	The operation is currently not applicable.

Value	Description
CBA_E_NOTROUTABLE	Routing is forbidden for this protocol and the remote station can't be reached via one of the local subnets.
CBA_E_OUTOFACCOPAIRS	The request exceeds the resources according to the ACCOPairs.
CBA_E_OUTOFPARTNERACCOS	The request exceeds the resources according to the partner ACCOs.
CBA_E_PERSISTRUNNING	While the Save service is running, no changes are allowed to the configuration data base (try again in a few seconds).
CBA_E_QCNOTAPPLICABLE	The quality code is not applicable for this operation.
CBA_E_QOSTYPENOTAPPLICABLE	The QoS type is not applicable for this operation.
CBA_E_QOSTYPEUNSUPPORTED	The QoS type is unsupported.
CBA_E_QOSVALUEUNSUPPORTED	The QoS value is unsupported.
CBA_E_SIZEEXCEEDED	The request exceeds the resources according to their size.
CBA_E_STATIONFAILURE	The remote station failed regarding the data transfer.
CBA_E_SUBELEMENTMISMATCH	The sub element access list does not match the item type.
CBA_E_TIMEVALUEUNSUPPORTED	The time value is unsupported.
CBA_E_TYPEMISMATCH	The type specified does not match the expected type.
CBA_E_UNKNOWNMEMBER	The member specified in the item identifier is not known.
CBA_E_UNKNOWNOBJECT	The object specified in the item identifier is not known.
CBA_S_ESTABLISHING	The connection is not yet established.
CBA_S_FRAMEEMPTY	RT Frame is empty, i.e. carries no connections.
CBA_S_NOCONNECTION	There is no connection from the provider to this specific consumer.
CBA_S_NOCONNECTIONDATA	Connection data is currently not received.
CBA_S_PERSISTPENDING	The value requested is currently not persistent.
CBA_S_VALUEBUFFERED	The value was only buffered and has no immediate effect on the process (e.g. device is in state CBAReady).
CBA_S_VALUEUNCERTAIN	It is uncertain, whether the value is valid, since the device is currently in state CBAReady.
DISP_E_BADINDEX	Invalid index.
DISP_E_BADPARAMCOUNT	The number of elements provided to DISPPARAMS is different from the number of arguments accepted by the method or property.
DISP_E_BADVARTYPE	One of the arguments is not a valid variant type.
DISP_E_EXCEPTION	The application needs to raise an exception. In this case, the structure passed in pExcepInfo shall be filled in.
DISP_E_MEMBERNOTFOUND	The requested member does not exist, or the call to Invoke tried to set the value of a read-only property.
DISP_E_NONAMEDARGS	This implementation of the Dispatch interface does not support named arguments
DISP_E_OVERFLOW	One of the arguments could not be coerced to the specified type.
DISP_E_PARAMNOTFOUND	One of the parameter DISPIDs does not correspond to a parameter on the method. In this case, puArgErr shall be set to the argument in which the error was detected.
DISP_E_PARAMNOTOPTIONAL	A required parameter was omitted.
DISP_E_TYPEMISMATCH	One or more of the arguments could not be coerced. The index within rgvarg of the first parameter with the incorrect type is returned in the puArgErr parameter.
DISP_E_UNKNOWNINTERFACE	The interface identifier passed in riid is not IID_NULL.
DISP_E_UNKNOWNLCID	Unknown language.
DISP_E_UNKNOWNNAME	Unknown name.
E_FAIL	Unspecified error.
E_INVALIDARG	One ore more arguments are invalid.
E_NOINTERFACE	No such interface supported.
E_NOTIMPL	The service is not implemented.
E_OUTOFMEMORY	Insufficient memory to complete the call.
RPC_E_INVALID_OBJECT	The requested object does not exist.
RPC_E_INVALID_OID	The specified object was not found or recognized.
RPC_E_INVALID_OXID	The object exporter was not found.
RPC_E_INVALID_SET	The object exporter set was not found.
RPC_E_VERSION_MISMATCH	The ORPC version on the client and server machine does not match.
RPC_S_PROCNUM_OUT_OF_RANGE	The procedure number is out of range.
S_FALSE	Success but with additional results, "function worked and the result is false".
S_OK	Success, "everything worked".
TYPE_E_ELEMENTNOTFOUND	Element not found.

5.3.1.7 Numeric types

5.3.1.7.1 Floating point types

5.3.1.7.1.1 float

This data type is the same as Float32.

5.3.1.7.1.2 Float32

CLASS:		**Data type**
ATTRIBUTES:		
1	Data type Numeric Identifier	= 8
2	Data type Name	= Float32
3	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.

5.3.1.7.1.3 double

This data type is the same as Float64.

5.3.1.7.1.4 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.

5.3.1.7.2 Fixed point types

5.3.1.7.2.1 Fixed point value E2

CLASS:		**Data type**
ATTRIBUTES:		
1	Data type Numeric Identifier	= 121
2	Data type Name	= E2
3	Format	= FIXED LENGTH
4.1	Octet Length	= 2

This data type is a signed linear fixed-point value with a length of two octets having a sign-bit, eigth binary places before the decimal point and seven binary places after it. The octets are coded according to Table 10.

Table 10 – E2 octets

Bit	8	7	6	5	4	3	2	1
Octet 1	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	sign	Integer						
Octet 2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
	inte-ger	Fraction						

The value range of E2 is shown in Table 11.

Table 11 – E2 value range

Range of values	Resolution
$-256+2^{-7}$ – $256-2^{-7}$	2^{-7} = 0,007 812 5

5.3.1.7.2.2 Unipolar2.16

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	125
2	Data type Name	=	Unipolar2.16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This data type is an unsigned linear fixed-point value with two octets having two binary places before the decimal point and fourteen binary places after it. The octets are coded according to Table 12.

Table 12 – Unipolar2.16 octets

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

The data type expresses a value in percent of a span. The value range of Unipolar2.16, which is defined in IEC 61375, is shown in Table 13.

Table 13 – Unipolar2.16 value range

Range of values	Resolution
0 % ≤ i ≤ $(400-2^{-14})$ %	2^{-14} = 0,006 1%

5.3.1.7.3 Integer types

5.3.1.7.3.1 char

This data type is the same as Integer8.

5.3.1.7.3.2 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.

5.3.1.7.3.3 short

This data type is the same as Integer16.

5.3.1.7.3.4 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.

5.3.1.7.3.5 long

This data type is the same as Integer32.

5.3.1.7.3.6 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.

5.3.1.7.3.7 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.

5.3.1.7.3.8 Normalized value N2

CLASS:		**Data type**	
ATTRIBUTES:			
1	Data type Numeric Identifier	=	113
2	Data type Name	=	N2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is the same as Integer16, i.e. a two's complement binary number with a length of two octets; however having a special interpretation.

The value provides a linear normalized value, such that the integer value 2^{14} (i.e. 16 384) equates to 100 %. The value is thus interpreted according to the following equation:

$$\text{Normalized Value N2 (in \%)} = \text{Value} / 16\,384 \quad (1)$$

The data type offers thus a resolution of 0,006 1 % (= 1 / 16 384). The N2 value range is shown in Table 14.

Table 14 – N2 value range

Value	Meaning
-32 768	-200 %
-32 767 .. -1	-199,993 896 484 375 % .. -0,006 1 %
-16 384	-100 %
0	0 %
16 384	100 %
1 .. 32 766	0,006 1 % .. 199,987 792 968 75 %
32 767	199,993 896 484 375 %

5.3.1.7.3.9 Normalized value N4

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	114
2	Data type Name	=	N4
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is the same as Integer32, i.e. a two's complement binary number with a length of four octets; however having a special interpretation.

The value provides a linear normalized value, such that the integer value 2^{30} (i.e. 1 073 741 824) equates to 100 %. The value is thus interpreted according to the following equation:

$$\text{Normalized Value N4 (in \%)} = \text{Value} / 1\,073\,741\,824 \quad (2)$$

The data type offers thus a resolution of approx. 0,000 000 093 1 % (= 1 / 1 073 741 824). The N4 value range is shown in Table 15.

Table 15 – N4 value range

Value	Meaning
−2 147 483 648	-200 %
−1 073 741 824	-100 %
−2 147 483 647.. -1	approx. -199,999 999 906 9 % .. -0,000 000 093 1 %
0	0 %
1 .. 2 147 483 646	approx. 0,000 000 093 1 % .. 199,999 999 813 7 %
1 073 741 824	100 %
2 147 483 647	approx. 199,999 999 906 9 %

5.3.1.7.3.10 Variable normalized X2

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	123
2	Data type Name	=	X2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is the same as Integer16, i.e. a two's complement binary number with a length of two octets; however having a special interpretation.

The value is interpreted as a linear normalized value (a value expressed as percentage). The interpretation is similar to the data type N2, however the normalization to 100 % is variable

and determined by an additional parameter “X” outside the scope of this data type, such that the integer value 2^X equates to 100 %.

NOTE 1 If the additional specified constant X equates to 14, the value range of X2 equates to that of the data type N2.

The value is interpreted according to the following equation:

$$\text{Normalized Value X2 (in \%)} = \text{Value} / 2^X \quad (3)$$

The data type offers thus a resolution of $1 / 2^X$ % or else 2^{-X}. Lower and upper bound of the value are computed according to the following equations:

$$\text{Lower Bound X2 (in \%)} = -2^{(15-X)} \quad (4)$$

$$\text{Upper Bound X2 (in \%)} = 2^{(15-X)} - 2^{-X} \quad (5)$$

The X2 value range for selected values of “X” is shown in Table 16.

Table 16 – X2 value range

Parameter “X”	Value	Meaning
	-32 768 .. 32 767	$-2^{(15-X)}$ % ≤ i ≤ $(2^{(15-X)} - 2^{-X})$ %
e.g. X = 10	-32 768 .. 32 767	-3200 % ≤ i ≤ -3199,90 %
e.g. X = 12	-32 768 .. 32 767	-800 % ≤ i ≤ 799,98 %
e.g. X = 14	-32 768 .. 32 767	-200 % ≤ i ≤ 199,99 %

NOTE 2 The value of this normalized value parameter refers to the additional specified constant X. The associated constant X is required in order to interpret the internal value.

5.3.1.7.3.11 Variable normalized X4

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	124
2	Data type Name	=	X4
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is the same as Integer32, i.e. a two’s complement binary number with a length of four octets, however having a special interpretation.

The value is interpreted a linear normalized value (a value expressed as percentage). The interpretation is similar to the data type N4, however the normalization to 100 % is variable and determined by an additional parameter “X” outside the scope of this data type, such that the integer value 2^X equates to 100 %.

NOTE 1 If the additional specified constant X equates to 30, the value range of X4 equates to that of the data type N4.

The value is interpreted according to the following equation:

$$\text{Normalized Value X4 (in \%)} = \text{Value} / 2^X \quad (6)$$

The data type offers thus a resolution of $1 / 2^X$ % or else 2^{-X}. Lower and upper bound of the value are computed according to the following equations:

$$\text{Lower Bound X4 (in \%)} = -2^{(31-X)} \quad (7)$$

$$\text{Upper Bound X4 (in \%)} = 2^{(31-X)} - 2^{-X} \quad (8)$$

The X4 value range for selected values of "X" is shown in Table 16.

Table 17 – X4 value range

Parameter "X"	Value	Meaning
	−2 147 483 648 .. 2 147 483 647	$-2^{(31-X)}$ % ≤ i ≤ ($2^{(31-X)} - 2^{-X}$) %
e.g. X = 26	−2 147 483 648 .. 2 147 483 647	-3200 % ≤ i ≤ -3200 %
e.g. X = 28	−2 147 483 648 .. 2 147 483 647	-800 % ≤ i ≤ 800 %
e.g. X = 30	−2 147 483 648 .. 2 147 483 647	-200 % ≤ i ≤ 200 %

NOTE 2 The value of this normalized value parameter refers to the additional specified constant X. The associated constant X is required in order to interpret the internal value.

5.3.1.7.3.12 Fixed point value C4

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	122
2	Data type Name	=	C4
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type is the same as Integer32, i.e. a two's complement binary number with a length of four octets; however having a special interpretation.

The value is interpreted a linear fixed point value with four decimal places according to the following equation:

$$\text{Fixed Point Value C4} = \text{Value} / 10\,000 \quad (9)$$

The data type offers thus a resolution of 0,000 1 (= 1 / 10 000). The C4 value range is shown in Table 18.

Table 18 – C4 value range

Value	Meaning
-2 147 483 648	-214 748,3648
0	0,000
2 147 483 647	214 748,3647
-2 147 483 648 – 2 147 483 647	value / 10 000

5.3.1.7.4 Unsigned types

5.3.1.7.4.1 unsigned char

This data type is the same as Unsigned8.

5.3.1.7.4.2 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 byte 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.

5.3.1.7.4.3 unsigned short

This data type is the same as Unsigned16.

5.3.1.7.4.4 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 byte 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.

5.3.1.7.4.5 unsigned long

This data type is the same as Unsigned32.

5.3.1.7.4.6 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 byte 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.

5.3.1.7.4.7 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 byte 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.

5.3.1.7.5 Time constant T2

CLASS:	**Data type**		
ATTRIBUTES:			
1	Data type Numeric Identifier	=	118
2	Data type Name	=	T2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is the same as Unsigned16, i.e. a two's complement binary number with a length of two octets; however having a special interpretation.

The value provides a time value based on a defined sampling time "Ta" outside the scope of this data type.

The value is interpreted according to the following equation:

Time Value = Value * Ta (10)

The data type offers thus a resolution of “Ta”. The T2 value range is shown in Table 19. Note that the value range is restricted; values outside the result provide a time value of “0”.

Table 19 – T2 value range

Value	Meaning
0 – 32 767	Value * Ta
32 768 – 65 535	0

NOTE The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.7.6 Time constant T4

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	119
2	Data type Name	=	T4
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is the same as Unsigned32, i.e. a two’s complement binary number with a length of four octets; however having a special interpretation.

The value provides a time value based on a defined sampling time “Ta” outside the scope of this data type.

The value is interpreted according to the following equation:

Time Value = Value * Ta (11)

The data type offers thus a resolution of “Ta”. The T4 value range is shown in Table 20. In contrary to the data type T2 the value range is not restricted.

Table 20 – T2 value range

Value	Meaning
0 – 4 294 967 295	Value * Ta

NOTE The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.7.7 Time constant D2

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	120
2	Data type Name	=	D2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is the same as Unsigned16, i.e. a two’s complement binary number with a length of two octets; however having a special interpretation.

The value provides a time value based on a defined sampling time “Ta” outside the scope of this data type.

The value is interpreted according to the following equation:

$$\text{Time value} = \text{Value} \times (\text{Ta} / 16\,384) \quad (12)$$

The data type offers thus a fraction of the constant sampling time with a resolution of “Ta / 16 384”. The D2 value range is shown in Table 21. Note that the value range is restricted; values outside the result provide a time value of “0”.

Table 21 – D2 value range

Value	Meaning
0 – 32 767	Value * (Ta / 16 384)
32 768 – 65 535	0

NOTE The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.7.8 Time constant R2

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	117
2	Data type Name	=	R2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is the same as Unsigned16, i.e. a two’s complement binary number with a length of two octets; however having a special interpretation.

The value provides a time value based on a defined sampling time “Ta” outside the scope of this data type.

The value is interpreted according to the following equation:

$$\text{Time value} = 16\,384 * (\text{Ta} / \text{Value}) \quad (13)$$

The data type offers thus a reciprocal multiple of the constant sampling time with a resolution of “Ta / Value”. The R2 value range is shown in Table 21. Note that the value range is restricted; values outside the result provide a time value of “0”.

Table 22 – R2 value range

Value	Meaning
0	16 384
1 – 16 384	16 384 * (Value / Ta)
16 385 – 65 535	16 384

NOTE The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.8 OctetString character types

5.3.1.8.1 UUID

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	1 025
2	Data type Name	=	UUID
3	Format	=	STRUCTURE

5.1	Number of Fields	=	4
5.2.1	Field Name	=	Data1
5.2.2	Field Data type	=	unsigned long
5.2.3	Field Name	=	Data2
5.2.4	Field Data type	=	unsigned short
5.2.5	Field Name	=	Data3
5.2.6	Field Data type	=	unsigned short
5.2.7	Field Name	=	Data4
5.2.8.1	Format	=	ARRAY
5.2.8.4.1	Number of Array Elements	=	8
5.2.8.4.2	Array Element Data type	=	unsigned char

This data type defines a 16 octet fixed length data type. The semantic is specified by the used ORPC model and beyond the scope of this standard.

This data type is structured as follows.

Data1
This field contains the first eight hexadecimal digits of the UUID.

Data2
This field contains the first group of four hexadecimal digits of the UUID.

Data3
This field contains the second group of four hexadecimal digits of the UUID.

Data4
This field contains an array of eight elements. The first two elements specifiy the third group of four hexadecimal digits of the UUID. The remaining six elements specify the final 12 hexadecimal digits of the UUID.

Specific values for decentralized peripherals items are shown in Table 23.

Table 23 – UUID for decentralized peripherals

Value	Description
UUID_NIL 00000000-0000-0000-0000-000000000000	Default AR in a device
UUID_IO_ObjectInstance_XYZ DEA00000-6C97-11D1-8271-{xxxxyyyyzzzz}	Identifies a special object instance within a physical device in case there are more than one where - xxxx represents the instance or node number - yyyy identify the Device ID as a vendor specific number for the device class and - zzzz represents the Vendor ID as a central administrative number.
UUID_IO_DeviceInterface DEA00001-6C97-11D1-8271-00A02442DF7D	Identifies the IO device interface uniquely.
UUID_IO_ControllerInterface DEA00002-6C97-11D1-8271-00A02442DF7D	Identifies the IO controller interface uniquely.
UUID_IO_SupervisorInterface DEA00003-6C97-11D1-8271-00A02442DF7D	Identifies the IO supervisor interface uniquely.
UUID_IO_ParameterServerInterface DEA00004-6C97-11D1-8271-00A02442DF7D	Identifies the IO parameter server interface uniquely.

Predefined values for distributed automation items are shown in Table 24.

Table 24 – UUID for distributed automation

Value	Description
UUID_NULL	
UUID_IUnknown	Identifies the IUnknown interface uniquely.
UUID_IDispatch	Identifies the IDispatch interface uniquely.
UUID_ICBAPhysicalDevice	Identifies the ICBAPhysicalDevice interface uniquely.
UUID_ICBAPhysicalDevice2	Identifies the ICBAPhysicalDevice2 interface uniquely.
UUID_ICBABrowse	Identifies the ICBABrowse interface uniquely.
UUID_ICBABrowse2	Identifies the ICBABrowse2 interface uniquely.
UUID_ICBAPersist	Identifies the ICBAPersist interface uniquely.
UUID_ICBAPersist2	Identifies the ICBAPersist2 interface uniquely.
UUID_ICBALogicalDevice	Identifies the ICBALogicalDevice interface uniquely.
UUID_ICBALogicalDevice2	Identifies the ICBALogicalDevice2 interface uniquely.
UUID_ICBAState	Identifies the ICBAState interface uniquely.
UUID_ICBATime	Identifies the ICBATime interface uniquely.
UUID_ICBAGroupError	Identifies the ICBAGroupError interface uniquely.
UUID_ICBAAccoMgt	Identifies the ICBAAccoMgt interface uniquely.
UUID_ICBAAccoMgt2	Identifies the ICBAAccoMgt2 interface uniquely.
UUID_ICBAAccoServer	Identifies the ICBAAccoServer interface uniquely.
UUID_ICBAAccoServer2	Identifies the ICBAAccoServer2 interface uniquely.
UUID_ICBAAccoServerSRT	Identifies the ICBAAccoServerSRT interface uniquely.
UUID_ICBAAccoCallback	Identifies the ICBAAccoCallback interface uniquely.
UUID_ICBAAccoCallback2	Identifies the ICBAAccoCallback2 interface uniquely.
UUID_ICBAAccoSync	Identifies the ICBAAccoSync interface uniquely.
UUID_ICBARTAuto	Identifies the ICBARTAuto interface uniquely.
UUID_ICBARTAuto2	Identifies the ICBARTAuto2 interface uniquely.
UUID_ICBASystemProperties	Identifies the ICBASystemProperties interface uniquely.
UUID_PhysicalDevice	Identifies the Physical device class uniquely.
UUID_LogicalDevice	Identifies the Logical Device class uniquely.
UUID_ACCO	Identifies the ACCO class uniquely.
UUID_RTAuto	Identifies the RT-Auto class uniquely.
UUID_SystemRTAuto	Identifies the System RT-Auto class uniquely.

5.3.1.9 Pointer types

5.3.1.9.1 Interface Pointer

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2057
2	Data type Name	=	Interface Pointer
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type defines a 4 octet fixed length data type.

5.3.1.9.2 LPWSTR

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2058
2	Data type Name	=	LPWSTR
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type defines a reference to a UnicodeString.

5.3.1.10 Time types

5.3.1.10.1 NetworkTime

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	58
2	Data type Name	=	NetworkTime
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This data type is based on the IETF RFC 1305 standard and composed of two unsigned values that express the network time related to a particular date (see Figure 2).

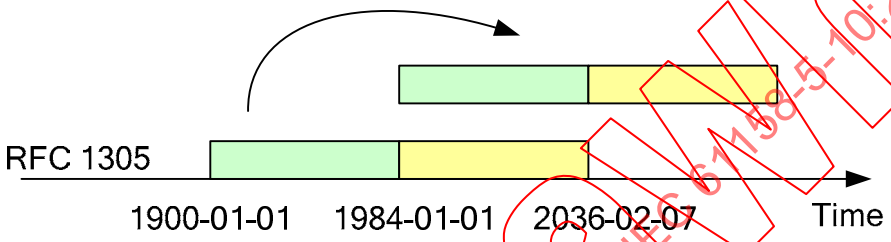


Figure 2 – NetworkTime date relation

The first element is an Unsigned32 data type that provides the network time in seconds since 1900-01-01 T 00:00:00 (UTC) or since 2036-02-07 T 06:28:16 (UTC) for time values less than 0x9DFF4400, which represents the 1984-01-01 T 00:00:00 (UTC). The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s. Rollovers after 136 years are not automatically detectable and are to be maintained by the application (see Table 25 and Table 26).

Table 25 – NetworkTime values

Offset	Value range	Resolution
Octet 1 to 4	0 – (2^{32}-1)	S
Octet 5 to 8	0 – (2^{32}-1)	$1/2^{32}$ s

Table 26 – NetworkTime octets

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of seconds since 1900-01-01. Rollover after 136 years. Thus, next would be 2036-02-07.
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fractional portion of seconds: $1/2^{32}$ s
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

The least significant Bit of the fractional portion (2^0) is device internally used to indicate a synchronized or unsynchronized state of the clock time.

5.3.1.10.2 NetworkTimeDifference

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	59

2	Data type Name	=	NetworkTimeDifference
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This data type is composed of an integer value and of an unsigned value that express the difference in network time. The first element is an Integer32 data type that provides the network time difference in seconds. The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s.

5.3.1.11 VisibleString character types

NOTE intentionally blank

5.3.2 String types

5.3.2.1 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 61158–6–3 and IEC 61158–6–10 define the order of transmission.

5.3.2.2 VisibleString

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	9
2	Data type Name	=	VisibleString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

This type is defined as the ISO/IEC 646 International Reference Version string type without the "del" (coding 0x7F) character.

5.3.2.3 UNICODEString

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	39
2	Data type Name	=	UNICODEString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

This type is defined as the ISO/IEC 10646 string type.

The Unicode Standard is the universal character encoding standard used for representation of text for computer processing. Versions of the Unicode Standard are fully compatible and synchronized with the corresponding versions of International Standard ISO/IEC 10646. UTF-8 is popular for HTML and similar protocols. UTF-8 is a way of transforming all Unicode characters into a variable length encoding of octets (see Table 27 and Table 28).

Table 27 – UNICODEString values

Numeric Identifier	Data type name	Value range	Resolution	Length
39	UNICODEString	0 to 0x0010FFFF	-	Variable; one character encoded into 1 up to 4 octets using UTF-8 encoding scheme (Table 28)

NOTE 1 The advantage of UTF-8 is its compatibility with the popular ASCII character set. Corresponding Unicode characters transformed into UTF-8 can easily be processed by many existing software packages without extensive software changes.

NOTE 2 The Unicode® Consortium is a registered trademark, and Unicode™ is a trademark of Unicode, Inc. The Unicode logo is a trademark of Unicode, Inc., and may be registered in some jurisdictions.

Table 28 – UTF-8 character encoding scheme

Unicode plane	UTF-8 Coding	Remark	Possibilities	
0000 0000– 0000 007F	0xxxxxxx	In this plane (128 characters) UTF-8 corresponds to the ASCII code; the most significant Bit is "0", the remaining 7 Bit are representing the ASCII character.	2^7	128
0000 0080– 0000 07FF	110xxxxx 10xxxxxx	The first octet contains 11xxxxxx, the following octets 10xxxxxx; x representing the Bit combination of a particular Unicode character. The number of "1" before the first "0" within the first byte is indicating the total number of octets for the particular character. Numbers in brackets are showing the theoretically possible characters.	$2^{11} - 2^7$ (2^{11})	1 920 (2 048)
0000 0800– 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx		$2^{16} - 2^{11}$ [2^{16}]	63 488 [65 536]
0001 0000– 0010 FFFF [0001 0000– 001F FFFF]	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx		2^{20} [2^{21}]	1 048 576 [2 097 152]

5.3.3 Structure types

5.3.3.1 ADDCONNECTIONIN

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2059
2	Data type Name	=	ADDCONNECTIONIN
3	Format	=	STRUCTURE
5.1	Number of Fields	=	5
5.2.1	Field Name	=	ProviderItem
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	ConsumerItem
5.2.4	Field Data type	=	LPWSTR
5.2.5	Field Name	=	Persistence
5.2.6	Field Data type	=	PERSISTDEF
5.2.7	Field Name	=	SubstituteValue
5.2.8	Field Data type	=	VARIANT
5.2.9	Field Name	=	Epsilon
5.2.10	Field Data type	=	VARIANT

This data type defines the ADDCONNECTIONIN data type.

This data type is structured as follows.

ProviderItem
This field contains the name of the source data item.

ConsumerItem
This field contains the name of the sink data item.

Persistence
This field describes the required persistence of the connection information. The values CBAVolatile and CBAPersistent are allowed.

SubstituteValue
This field contains the substitute value according the data type of the connection item.

Epsilon
This field specifies the hysteresis as absolute change of the value.

5.3.3.2 ADDCONNECTIONOUT

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2060
2	Data type Name	=	ADDCONNECTIONOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	3
5.2.1	Field Name	=	ConsumerID
5.2.2	Field Data type	=	unsigned long
5.2.3	Field Name	=	Version
5.2.4	Field Data type	=	unsigned short
5.2.5	Field Name	=	ErrorState
5.2.6	Field Data type	=	HRESULT

This data type defines the ADDCONNECTIONOUT data type.

This data type is structured as follows.

ConsumerID
This field contains the identifier of the consumer.

Version
This field contains the version number of the connection.

ErrorState
This field contains the error condition of the connection.

5.3.3.3 BSTR

CLASS:	**Data type**		
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2061
2	Data type Name	=	BSTR
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	ByteCount
5.2.2	Field Data type	=	unsigned long
5.2.3	Field Name	=	UnicodeString
5.2.4	Field Data type	=	UnicodeString

This data type defines an UnicodeString with preceding byte count value. The count contains the number of octets, not UNICODE characters, in the string. This count does not include the terminating zero character (2 octets).

5.3.3.4 CONNECTIN

CLASS:	**Data type**		
ATTRIBUTES:			
1	Data type Numeric Identifier	=	2062
2	Data type Name	=	CONNECTIN

3	Format	=	STRUCTURE
5.1	Number of Fields	=	4
5.2.1	Field Name	=	ProviderItem
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	DataType
5.2.4	Field Data type	=	VARTYPE
5.2.5	Field Name	=	Epsilon
5.2.6	Field Data type	=	VARIANT
5.2.7	Field Name	=	ConsumerID
5.2.8	Field Data type	=	unsigned long

This data type defines the CONNECTIN data type.

This data type is structured as follows.

ProviderItem
This field contains the name of the source data item.

DataType
This field contains the type code for the data item.

Epsilon
This field contains the hysteresis as absolute change of the value.

ConsumerID
This field contains the identifier of the consumer.

5.3.3.5 CONNECTIN2

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2063
2	Data type Name	=	CONNECTIN2
3	Format	=	STRUCTURE
5.1	Number of Fields	=	5
5.2.1	Field Name	=	ProviderItem
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	TypeDescLen
5.2.4	Field Data type	=	unsigned char
5.2.5	Field Name	=	pTypeDesc
5.2.6	Field Data type	=	unsigned long
5.2.7	Field Name	=	Epsilon
5.2.8	Field Data type	=	VARIANT
5.2.9	Field Name	=	ConsumerID
5.2.10	Field Data type	=	unsigned long

This data type defines the CONNECTIN2 data type.

This data type is structured as follows.

ProviderItem
This field contains the name of the source data item.

TypeDescLen
This field contains the length of the extended type description for the data item.

pTypeDesc
This field contains the reference to the extended type description for the data item. It is built according to the rules described in 7.2.3.

Epsilon
This field contains the hysteresis as absolute change of the value.

ConsumerID
This field contains the identifier of the consumer.

5.3.3.6 CONNECTINCR

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2064
2	Data type Name	=	CONNECTINCR
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	ConsumerCRID
5.2.2	Field Data type	=	unsigned short
5.2.3	Field Name	=	ConsumerCRLength
5.2.4	Field Data type	=	unsigned short

This data type defines the CONNECTINCR data type.

This data type is structured as follows.

ConsumerCRID
This field contains the frame ID of the CR.

ConsumerCRLength
This field contains the maximum length of the CR.

5.3.3.7 CONNECTINSRT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2065
2	Data type Name	=	CONNECTINSRT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	5
5.2.1	Field Name	=	ProviderItem
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	TypeDescLen
5.2.4	Field Data type	=	unsigned char
5.2.5	Field Name	=	pTypeDesc
5.2.6	Field Data type	=	unsigned long
5.2.7	Field Name	=	ConsumerID
5.2.8	Field Data type	=	unsigned long
5.2.9	Field Name	=	Length
5.2.10	Field Data type	=	unsigned short

This data type defines the CONNECTINSRT data type.

This data type is structured as follows.

ProviderItem
This field contains the name of the source data item.

TypeDescLen
This field contains the length of the extended type description for the data item.

pTypeDesc
This field contains the reference to the extended type description for the data item. It is built according to the rules described in 7.2.3.

ConsumerID
This field contains the consumer ID.

Length
This field contains the marshalled length of the transfered data.

5.3.3.8 CONNECTOUT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2066
2	Data type Name	=	CONNECTOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	ProviderID
5.2.2	Field Data type	=	unsigned long
5.2.3	Field Name	=	ErrorState
5.2.4	Field Data type	=	HRESULT

This data type defines the CONNECTOUT data type.

This data type is structured as follows.

ProviderID
This field contains the identifier of the provider.

ErrorState
This field contains the error condition of the connection.

5.3.3.9 CONNECTOUTCR

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2067
2	Data type Name	=	CONNECTOUTCR
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	ProviderCRID
5.2.2	Field Data type	=	unsigned short
5.2.3	Field Name	=	PartialResult
5.2.4	Field Data type	=	unsigned short

This data type defines the CONNECTOUTCR data type.

This data type is structured as follows.

ProviderCRID
This field contains the provider CR ID.

PartialResult
This field contains the partial result.

5.3.3.10 DIAGCONSCONNOUT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2068
2	Data type Name	=	DIAGCONSCONNOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	5
5.2.1	Field Name	=	State
5.2.2	Field Data type	=	VARIANT_BOOL
5.2.3	Field Name	=	Persistence
5.2.4	Field Data type	=	PERSISTDEF
5.2.5	Field Name	=	Version
5.2.6	Field Data type	=	unsigned short
5.2.7	Field Name	=	ErrorState
5.2.8	Field Data type	=	HRESULT
5.2.9	Field Name	=	PartialResult
5.2.10	Field Data type	=	HRESULT

This data type defines the DIAGCONSCONNOUT data type.

This data type is structured as follows.

State
This field contains the state of the connection. The value TRUE indicates an active and the value FALSE an inactive connection.

Persistence
This field describes the required persistence of the connection information.

Version
This field contains the version number of the connection.

ErrorState
This field contains the error condition of the connection.

PartialResult
This field contains the partial result.

5.3.3.11 DISPPARAMS

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2069
2	Data type Name	=	DISPPARAMS
3	Format	=	STRUCTURE
5.1	Number of Fields	=	4
5.2.1	Field Name	=	rgvarg
5.2.2.1	Format	=	ARRAY
5.2.2.4.1	Number of Array Elements	=	cArgs
5.2.2.4.2	Array Element Data type	=	VARIANT
5.2.3	Field Name	=	rgdispidNamedArgs
5.2.4.1	Format	=	ARRAY
5.2.4.4.1	Number of Array Elements	=	cNamedArgs
5.2.4.4.2	Array Element Data type	=	long
5.2.5	Field Name	=	cArgs
5.2.6	Field Data type	=	unsigned short
5.2.7	Field Name	=	cNamedArgs
5.2.8	Field Data type	=	unsigned short

This data type defines the DISPPARAMS data type used to contain the parameters passed to a service.

This data type is structured as follows.

rgvarg
This field contains the list of arguments.

rgdispidNamedArgs
This field contains the Dispatch IDs of named arguments.

cArgs
This field contains the number of arguments.

cNamedArgs
This field contains the number of named arguments.

5.3.3.12 EXCEPINFO

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2070
2	Data type Name	=	EXCEPINFO
3	Format	=	STRUCTURE

5.1	Number of Fields	=	9
5.2.1	Field Name	=	wCode
5.2.2	Field Data type	=	unsigned short
5.2.3	Field Name	=	wReserved
5.2.4	Field Data type	=	unsigned short
5.2.5	Field Name	=	bstrSource
5.2.6	Field Data type	=	BSTR
5.2.7	Field Name	=	bstrDescription
5.2.8	Field Data type	=	BSTR
5.2.9	Field Name	=	bstrHelpFile
5.2.10	Field Data type	=	BSTR
5.2.11	Field Name	=	dwHelpContext
5.2.12	Field Data type	=	unsigned long
5.2.13	Field Name	=	pvReserved
5.2.14	Field Data type	=	unsigned long
5.2.15	Field Name	=	pfnDeferredFillIn
5.2.16	Field Data type	=	unsigned long
5.2.17	Field Name	=	scode
5.2.18	Field Data type	=	long

This data type defines the EXCEPINFO data type to describe an exception that occurred during the Invoke service.

This data type is structured as follows:

wCode
This field contains an error code identifying the error. Error codes shall be greater than 1 000. Either this field or the scode field shall be filled in; the other shall be set to 0.

wReserved
This field is reserved and shall be set to 0.

bstrSource
This field contains a textual, human-readable name of the source of the exception. Typically, this is an application name. This field shall be filled in by the implementor of the Dispatch interface.

bstrDescription
This field contains a textual, human-readable description of the error intended for the customer. If no description is available, use Null.

bstrHelpFile
This field contains the fully qualified drive, path, and file name of a Help file with more information about the error. If no Help is available, use Null.

dwHelpContext
This field contains the Help context ID of the topic within the Help file. This field shall be filled in if and only if the bstrHelpFile field is not Null.

pvReserved
This field is reserved and shall be set to Null.

pfnDeferredFillIn
This field contains the address of a function that takes an EXCEPINFO structure as an argument and returns an HRESULT value. If deferred, fill-in is not desired, this field shall be set to Null.

scode
This field contains a return value describing the error. Either this field or the wCode field shall be filled in; the other shall be set to 0.

5.3.3.13 FILETIME

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2071
2	Data type Name	=	FILETIME
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	LowDataTime
5.2.2	Field Data type	=	unsigned long
5.2.3	Field Name	=	HighDataTime
5.2.4	Field Data type	=	unsigned long

This data type is a 64 bit time value representing the number of 100-nanosecond intervals since 1601-01-01 00:00.

5.3.3.14 GETIDOUT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2072
2	Data type Name	=	GETIDOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	4
5.2.1	Field Name	=	ConsumerID
5.2.2	Field Data type	=	unsigned long
5.2.3	Field Name	=	State
5.2.4	Field Data type	=	VARIANT_BOOL
5.2.5	Field Name	=	Version
5.2.6	Field Data type	=	unsigned short
5.2.7	Field Name	=	ErrorState
5.2.8	Field Data type	=	HRESULT

This data type defines the GETIDOUT data type.

This data type is structured as follows.

ConsumerID
This field contains the identifier of the consumer.

State
This field contains the state of the connection. The value TRUE indicates an active and the value FALSE an inactive connection.

Version
This field contains the version number of the connection.

ErrorState
This field contains the error condition of the connection.

5.3.3.15 GETCONNECTIONOUT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2073
2	Data type Name	=	GETCONNECTIONOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	11
5.2.1	Field Name	=	Provider
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	ProviderItem
5.2.4	Field Data type	=	LPWSTR
5.2.5	Field Name	=	ConsumerItem
5.2.6	Field Data type	=	LPWSTR
5.2.7	Field Name	=	SubstituteValue

5.2.8	Field Data type	=	VARIANT
5.2.9	Field Name	=	Epsilon
5.2.10	Field Data type	=	VARIANT
5.2.11	Field Name	=	QoSType
5.2.12	Field Data type	=	unsigned short
5.2.13	Field Name	=	QoSValue
5.2.14	Field Data type	=	unsigned short
5.2.15	Field Name	=	State
5.2.16	Field Data type	=	VARIANT_BOOL
5.2.17	Field Name	=	Persistence
5.2.18	Field Data type	=	PERSISTDEF
5.2.19	Field Name	=	Version
5.2.20	Field Data type	=	unsigned short
5.2.21	Field Name	=	ErrorState
5.2.22	Field Data type	=	HRESULT

This data type defines the GETCONNECTIONOUT data type.

This data type is structured as follows.

Provider
This field contains the name of the providers LDev (Format: PDev!LDev).

ProviderItem
This field contains the name of the source data item.

ConsumerItem
This field contains the name of the sink data item.

SubstituteValue
This field specifies the substitute value according the data type of the connection item.

Epsilon
This field specifies the hysteresis as absolute change of the value.

QoSType
This field contains the quality of service type.

QoSValue
This field contains the quality of service value.

State
This field contains the state of the connection. The value TRUE indicates an active and the value FALSE an inactive connection.

Persistence
This field describes the required persistence of the connection information. The values CBAVolatile and CBAPersistent are allowed.

Version
This field contains the version number of the connection.

ErrorState
This field contains the error condition of the connection.

5.3.3.16 GETCONSCONNOUT

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2074
2	Data type Name	=	GETCONSCONNOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	10
5.2.1	Field Name	=	Provider
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	ProviderItem

5.2.4	Field Data type	=	LPWSTR
5.2.5	Field Name	=	ConsumerItem
5.2.6	Field Data type	=	LPWSTR
5.2.7	Field Name	=	SubstituteValue
5.2.8	Field Data type	=	VARIANT
5.2.9	Field Name	=	Epsilon
5.2.10	Field Data type	=	VARIANT
5.2.11	Field Name	=	QoSType
5.2.12	Field Data type	=	unsigned short
5.2.13	Field Name	=	QoSValue
5.2.14	Field Data type	=	unsigned short
5.2.15	Field Name	=	State
5.2.16	Field Data type	=	VARIANT_BOOL
5.2.17	Field Name	=	Persistence
5.2.18	Field Data type	=	PERSISTDEF
5.2.19	Field Name	=	PartialResult
5.2.20	Field Data type	=	HRESULT

This data type defines the GETCONSCONNOUT data type.

This data type is structured as follows.

Provider
This field contains the name of the providers LDev (Format: PDev!LDev).

ProviderItem
This field contains the name of the source data item.

ConsumerItem
This field contains the name of the sink data item.

SubstituteValue
This field specifies the substitute value according the data type of the connection item.

Epsilon
This field specifies the hysteresis as absolute change of the value.

QoSType
This field contains the quality of service type.

QoSValue
This field contains the quality of service qualifier.

State
This field contains the state of the connection. The value TRUE indicates an active and the value FALSE an inactive connection.

Persistence
This field describes the required persistence of the connection information. The values CBAVolatile and CBAPersistent are allowed.

PartialResult
This field contains the partial result.

5.3.3.17 GETPROVCONNOUT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2075
2	Data type Name	=	GETPROVCONNOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	8
5.2.1	Field Name	=	Consumer
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	ProviderItem

5.2.4	Field Data type	=	LPWSTR
5.2.5	Field Name	=	ConsumerID
5.2.6	Field Data type	=	unsigned long
5.2.7	Field Name	=	Epsilon
5.2.8	Field Data type	=	VARIANT
5.2.9	Field Name	=	QoSType
5.2.10	Field Data type	=	unsigned short
5.2.11	Field Name	=	QoSValue
5.2.12	Field Data type	=	unsigned short
5.2.13	Field Name	=	State
5.2.14	Field Data type	=	VARIANT_BOOL
5.2.15	Field Name	=	PartialResult
5.2.16	Field Data type	=	HRESULT

This data type defines the GETPROVCONNOUT data type.

This data type is structured as follows.

Consumer
This field contains the name of the consumers LDev (Format: PDev!LDev).

ProviderItem
This field contains the name of the source data item.

ConsumerID
This field specifies the consumer ID.

Epsilon
This field specifies the hysteresis as absolute change of the value.

QoSType
This field contains the quality of service type.

QoSValue
This field contains the quality of service value.

State
This field contains the state of the connection. The value TRUE indicates an active and the value FALSE an inactive connection.

PartialResult
This field contains the partial result.

5.3.3.18 MACAddr

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2076
2	Data type Name	=	MACAddr
3	Format	=	STRUCTURE
5.1	Number of Fields	=	6
5.2.1	Field Name	=	B0
5.2.2	Field Data type	=	unsigned char
5.2.3	Field Name	=	B1
5.2.4	Field Data type	=	unsigned char
5.2.5	Field Name	=	B2
5.2.6	Field Data type	=	unsigned char
5.2.7	Field Name	=	B3
5.2.8	Field Data type	=	unsigned char
5.2.9	Field Name	=	B4
5.2.10	Field Data type	=	unsigned char
5.2.11	Field Name	=	B5
5.2.12	Field Data type	=	unsigned char

This data type defines a 6 byte Ethernet MAC address (highest byte first).

5.3.3.19 OctetString2+Unsigned8

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	103
2	Data type Name	=	OctetString2+Unsigned8
5	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2	List of Fields		
5.2.1	Field Name	=	Value
5.2.2	Field Data type	=	Octet String(2)
5.2.3	Field Name	=	Status
5.2.4	Field Data type	=	Unsigned8

This data type defines the OctetString2+Unsigned8 and is coded according to Table 29.

Table 29 – OctetString2+Unsigned8 octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. octet							
Octet 2	2. octet							
Octet 3	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

This data type is structured as follows.

Value
This field contains the value as Octet String with the length of 2.

Status
This field describes the status of the value as a qualifier.

5.3.3.20 Float32+Unsigned8

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	101
2	Data type Name	=	Float32+Unsigned8
5	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2	List of Fields		
5.2.1	Field Name	=	Value
5.2.2	Field Data type	=	Float32
5.2.3	Field Name	=	Status
5.2.4	Field Data type	=	Unsigned8

This data type defines the Float32+Unsigned8 and is coded according to Table 30.

SN: sign 0 = positive, 1 = negative

Table 30 – Float32+Unsigned8 octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Exponent (E)							
	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
Octet 2	(E)	Fraction (F)						
	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Octet 3	Fraction (F)							
	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
Octet 4	Fraction (F)							
	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}
Octet 5	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

This data type is structured as follows.

Value
This field contains the value as Float32.

Status
This field describes the status of the value as a qualifier.

5.3.3.21 Unsigned8+Unsigned8

CLASS: **Data type**

ATTRIBUTES:

2	Data type Numeric Identifier	=	102
2	Data type Name	=	Unsigned8+Unsigned8
5	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2	List of Fields		
5.2.1	Field Name	=	Value
5.2.2	Field Data type	=	Unsigned8
5.2.3	Field Name	=	Status
5.2.4	Field Data type	=	Unsigned8

This data type defines the Unsigned8+Unsigned8 and is coded according to Table 31.

Table 31 – Unsigned8+Unsigned8 octets

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

This data type is structured as follows.

Value
This field contains the value as Unsigned8.

Status
This field describes the status of the value as a qualifier.

5.3.3.22 READITEMOUT

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	2077
2	Data type Name	=	READITEMOUT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	4
5.2.1	Field Name	=	Value
5.2.2	Field Data type	=	VARIANT
5.2.3	Field Name	=	QualityCode

5.2.4	Field Data type	=	ITEMQUALITYDEF
5.2.5	Field Name	=	TimeStamp
5.2.6	Field Data type	=	FILETIME
5.2.7	Field Name	=	ErrorState
5.2.8	Field Data type	=	HRESULT

This data type defines the READITEMOUT data type.

This data type is structured as follows.

Value
This field contains the value itself with the format according to the appropriate data type.

QualityCode
This field contains the Quality code of the data value.

TimeStamp
This field contains the time stamp of the value.

ErrorState
This field contains the error condition of the connection.

5.3.3.23 SAFEARRAY

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2078
2	Data type Name	=	RGSABOUND
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Elements
5.2.2	Field Data type	=	unsigned long
5.2.3	Field Name	=	Left Bound
5.2.4	Field Data type	=	long

This data type is a helper describing one array element with boundary information needed to define the SAFEARRAY below.

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2079
2	Data type Name	=	SAFEARRAY
3	Format	=	STRUCTURE
5.1	Number of Fields	=	6
5.2.1	Field Name	=	Dims
5.2.2	Field Data type	=	unsigned short
5.2.3	Field Name	=	Features
5.2.4	Field Data type	=	unsigned short
5.2.5	Field Name	=	Elements
5.2.6	Field Data type	=	unsigned long
5.2.7	Field Name	=	Locks
5.2.8	Field Data type	=	unsigned long
5.2.9	Field Name	=	Data Address
5.2.10	Field Data type	=	unsigned long
5.2.11	Field Name	=	Rgsabound
5.2.12.1	Format	=	ARRAY
5.2.12.4.1	Number of Array Elements	=	Dims
5.2.12.4.2	Array Element Data type	=	RGSABOUND

This data type expresses a one- or multi-dimensional array of a single data type. (However, this single data type can be a VARIANT, allowing you arrays of mixed types.) The reason

these arrays are called safe arrays is because they contain bounds information, allowing to check the index or indices against the bounds before accessing the data in the array. The lower bound of the array does not have to be zero, so the safe array has to store its lower bound as well as the size. Finally, safe arrays allow locking (and unlocking) so it can be sure the pointer to the data you get is valid.

This data type is structured as follows.

Dims
This field contains the dimension of the SAFEARRAY.

Features
This field contains flags for allocation type and data type of the SAFEARRAY.

Elements
This field contains the size of a single element of the SAFEARRAY.

Locks
This field contains the lock counter of the SAFEARRAY.

Data Address
This field contains the address of the actual data of the SAFEARRAY.

Rgsabound
This field contains an array with the information about boundaries for each dimension of the SAFEARRAY. Therefore, the number of array elements is according to the dimension of the safe array

5.3.3.24 VARIANT

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	2080
2	Data type Name	=	VARIANT
3	Format	=	STRUCTURE
5.1	Number of Fields	=	5
5.2.1	Field Name	=	Tag
5.2.2	Field Data type	=	VARTYPE
5.2.3	Field Name	=	Padding1
5.2.4	Field Data type	=	unsigned short
5.2.5	Field Name	=	Padding2
5.2.6	Field Data type	=	unsigned short
5.2.7	Field Name	=	Padding3
5.2.8	Field Data type	=	unsigned short
5.2.9	Field Name	=	Value
5.2.10	Field Data type	=	see Table 32

This data type expresses a VARIANT containing possible values of data types according to Table 32. The Tag field contains the numeric identifier of the data type to identify the data type of the value. The overall size of the VARIANT is 16 octets.

Table 32 – Data Types for Value in a VARIANT

Data type
VARIANT_BOOL
char
short
long
unsigned char
unsigned short
unsigned long
float
double
date
Address of a BSTR
Address of a SAFEARRAY
Address of a userdefined struct
Interface Pointer
HRESULT

5.3.3.25 WRITEITEMIN

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2081
2	Data type Name	=	WRITEITEMIN
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Item
5.2.2	Field Data type	=	LPWSTR
5.2.3	Field Name	=	Value
5.2.4	Field Data type	=	VARIANT

This data type defines the WRITEITEMIN data type.

This data type is structured as follows.

Item
This field contains the name of the data item.

Value
This field contains the value itself with the format according to the appropriate data type.

5.3.3.26 WRITEITEMQCDIN

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	2082
2	Data type Name	=	WRITEITEMQCDIN
3	Format	=	STRUCTURE
5.1	Number of Fields	=	3
5.2.1	Field Name	=	WriteItem
5.2.2	Field Data type	=	WRITEITEMIN
5.2.3	Field Name	=	QualityCode
5.2.4	Field Data type	=	ITEMQUALITYDEF
5.2.5	Field Name	=	TimeStamp
5.2.6	Field Data type	=	FILETIME

This data type defines the WRITEITEMQCDIN data type.

This data type is structured as follows.

WriteItem
This field contains the name and value of the data item.

QualityCode
This field contains the Quality code of the data value.

TimeStamp
This field contains the time stamp of the value.

5.3.3.27 Unsigned16_S

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	104
2	Data type Name	=	Unsigned16_S
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This data type is structured and coded according to Table 33 and Table 34.

Table 33 – Unsigned16_S octets

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}
Octets 2	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	St1	St0

Table 34 – Unsigned16_S meaning

St1 (bit1)	St0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.3.28 Integer16_S

CLASS: Data type
ATTRIBUTES:

1	Data type Numeric Identifier	=	105
2	Data type Name	=	Integer16_S
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This data type is structured and coded according to Table 35 and Table 36.

SN: sign 0 = positive, 1 = negative

Table 35 – Integer16_S octets

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}
Octets 2	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	St1	St0

Table 36 – Integer16_S meaning

St1 (bit1)	St0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.3.29 Unsigned8_S

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	106
2	Data type Name	=	Unsigned8_Status
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This data type is structured and coded according to Table 37 and Table 38.

Table 37 – Unsigned8_S octets

Bits	7	6	5	4	3	2	1	0
Octets 1	2^5	2^4	2^3	2^2	2^1	2^0	St1	St0

Table 38 – Unsigned8_S meaning

St1 (bit1)	St0 (bit0)	meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.3.30 OctetString_S

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	107
2	Data type Name	=	OctetString_S
3	Format	=	STRING
4.1	Octet Length	=	N

This data type is structured and coded according to Table 39 and Table 40.

Table 39 – OctetString_S octets

Bits	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Octet 1	ch(8)	ch(7)	ch(6)	ch(5)	ch(4)	ch(3)	ch(2)	ch(1)
***	***	***	***	***	***	***	***	***
Octet m	ch(n)	ch(n-1)	ch(n-2)	ch(n-3)	ch(n-4)	ch(n-5)	ch(n-6)	ch(n-7)
Octet m+1	st1(4)	st0(4)	st1(3)	st0(3)	st1(2)	st0(2)	st1(1)	st0(1)
***	***	***	***	***	***	***	***	***
Octet 3*m	st1(n)	st0(n)	st1(n-1)	st0(n-1)	st1(n-2)	st0(n-2)	st1(n-3)	st0(n-3)

ch(x) value for channel x; st(x) status information for channel x; 1<x≤n

Table 40 – OctetString_S status bits

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.3.31 F message trailer with 4 octets

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	110
2	Data type Name	=	F message trailer with 4 octets
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data structure consists of the status/control octet and 3 octets CRC parameters. This data type can be associated with input or output data up to 12 byte.

This data type is structured and coded according to Table 41.

Table 41 – F message trailer with 4 octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	High-octet CRC							
Octet 3	CRC							
Octet 4	Low- octet CRC (least significant octet)							

5.3.3.32 F message trailer with 5 octets

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	111
2	Data type Name	=	F message trailer with 5 octets
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	5

This data structure consists of the status/control octet and 4 octet CRC parameters. This data type can be associated with input or output data up to 122 octets.

This data type is structured and coded according to Table 42.

Table 42 – F message trailer with 5 octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	1. octet CRC (most significant octet)							
Octet 3	2. octet CRC							
Octet 4	3. octet CRC							
Octet 5	4. octet CRC (least significant octet)							

5.4 Data type ASE service specification

There are no operational services defined for the type object.

6 Communication model for common services

6.1 Concepts

The concepts of IEC/TR 61784–1, Clause 9 common elements are applied. This part of the specification defines common services, which are used for distributed automation and decentralized periphery. Furthermore, this part of the specification defines the usage of, refinements, or extensions to the referenced standards.

6.2 ASE data types

The data types supported by common services are a subset of those defined in Clause 5. They are:

Boolean
BinaryDate
TimeOfDay
TimeOfDay with date indication
TimeOfDay without date indication
TimeDifference
TimeDifference with date indication
Float32
Float64
Integer8
Integer16
Integer32
Integer64
Unsigned8
Unsigned16
Unsigned32
Unsigned64
UUID
NetworkTime
NetworkTimeDifference

OctetString

VisibleString

6.3 ASEs

6.3.1 Discovery and basic configuration ASE

6.3.1.1 Overview

The discovery and basic configuration is a concept of reading and writing basic device network configuration parameters and discovering devices by means of different filter criteria. It uses IEEE 802.3 services.

6.3.1.2 DCP class specification

6.3.1.2.1 General

The DCP ASE defines one DCP object type.

6.3.1.2.2 Template

A DCP object is described by the following template:

ASE:	**DCP ASE**		
CLASS:	**DCP**		
CLASS ID:	**not used**		
PARENT CLASS:	**IEEE 802.1AB**		
ATTRIBUTES:			
1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	IP
2.1	(m)	Attribute:	MAC Address
2.2	(m)	Attribute:	IP Parameter
2.2.1	(m)	Attribute:	IP Address
2.2.2	(m)	Attribute:	Subnet Mask
2.2.3	(m)	Attribute:	Standard Gateway
3.	(m)	Attribute:	Device Properties
3.1	(m)	Attribute:	Device ID
3.2	(m)	Attribute:	Device Role Details
3.3	(m)	Attribute:	Device Vendor
3.4	(m)	Attribute:	List of Device Options
3.4.1	(m)	Attribute:	Device Option
3.4.2	(m)	Attribute:	Device Suboption
3.5	(m)	Attribute:	Name Of Station
3.6	(m)	Attribute:	Alias Name Of Station
3.7	(m)	Attribute:	Device Initiative
4	(o)	Attribute:	DHCP
4.1	(m)	Attribute:	Host Name
4.2	(m)	Attribute:	Vendor Specific Information
4.3	(m)	Attribute:	Server Identifier
4.4	(m)	Attribute:	Parameter Request List
4.5	(m)	Attribute:	Class Identifier
4.6	(m)	Attribute:	DHCP Client Identifier
4.7	(m)	Attribute:	Fully Qualified Domian Name
4.8	(m)	Attribute:	UUID/GUID-based Client Identifier
4.9	(m)	Attribute:	Control DHCP for Address Resolution
5	(o)	Attribute:	Manufacturer Specific

5.1	(m)	Attribute:	List of Manufacturer Specific Suboptions
5.1.1	(m)	Attribute:	Manufacturer Specific Suboption
5.1.2	(m)	Attribute:	Manufacturer OUI
5.1.3	(m)	Attribute:	Manufacturer Specific String
6	(m)	Attribute:	Protocol machine Parameter
6.1	(m)	Attribute:	Max Retry Limit
6.2	(m)	Attribute:	UC Client Timeout
6.3	(m)	Attribute:	MC Client Timeout
6.4	(m)	Attribute:	TagControlInformation
6.4.1	(m)	Attribute:	VLAN ID
6.4.2	(m)	Attribute:	Priority
6.5	(m)	Attribute:	Client Hold Time

SERVICES:

1	(m)	OpsService:	Get
2	(m)	OpsService:	Set
3	(m)	OpsService:	Identify
4	(o)	OpsService:	Hello

6.3.1.2.3 Attributes

Implicit
The attribute Implicit indicates that the DCP object is implicitly addressed by the service.

IP
This attribute contains the following suboption attributes:

MAC Address
This attribute contains physical address of the device according to IEEE 802.3 MAC address.

The MAC Address shall be non volatile. It may be read by means of the Get service or it may be used within the Identify service as filter. It shall not be used as a suboption within the Set service.

Attribute type: OctetString[6]

IP Parameter
The IP Parameter shall have the same meaning as the corresponding attributes within the IP suite ASE as defined in 6.3.11. The service parameter Data Qualifier shall be used to address currently used or permanent stored attribute values for DCP services. This attribute list contains the following attributes:

IP Address
This attribute contains the non volatile IP address according to RFC 791 and RFC 3330.

Attribute type: Unsigned32

Default Value: 0.0.0.0

Subnet Mask
This attribute contains the non volatile subnet mask according to RFC 791 and RFC3 330.

Attribute type: Unsigned32

Default Value: 0.0.0.0

Standard Gateway
This attribute contains the non volatile IP address of the standard gateway according to RFC 791 and RFC3 330.

Attribute type: Unsigned32

Default Value: 0.0.0.0

Device Properties
This attribute contains the following suboption attributes:

Device ID
This attribute contains the Device Ident Number as described in 8.6.1.

Attribute type: Unsigned32

Device Role Details
This attribute contains the role of the device.

Attribute type: Unsigned8

Allowed values: IO_DEVICE, IO_CONTROLLER, IO_MULTIDEVICE, IO_SUPERVISOR

Device Vendor
This attribute contains a vendor specific string. It may be the type of the device or an ordering number.

Attribute type: OctetString

List of Device Options
This attribute list shall contain all options which are supported by the device:

Device Option
This attribute contains an option that is supported by the device.

Attribute type: Unsigned8

Allowed values: IP, DEVICE_PROPERTIES, DHCP, CONTROL, MANUFACTURER_SPECIFIC_128, ... MANUFACTURER_SPECIFIC_254

Device Suboption
This attribute contains a suboption related to a specific option that is supported by the device.

Attribute type: Unsigned8

Allowed values: SIGNAL, FACTORY_RESET, MAC_ADDRESS, IP_PARAMETER, MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255, NAME_OF_STATION, DEVICE_ID, DEVICE_ROLE, DEVICE_VENDOR, DEVICE_OPTIONS, ALIAS_NAME, DHCP_PARAMETER

Name Of Station
This attribute contains the name of the station provided by engineering. The value of this attribute shall be the same as for the attribute Chassis ID of the IEEE 802.1AB ASE. The values “port-xyz” or “port-xyz-rstuv” where x,y,z is in the range “0”-“9” from 001 up to 255 and r, s, t, u, v is in the range “0”-“9” from 00 000 up to 65 535 shall not be used.

Station Name Alias
This attribute contains the alias name of the station derived from LLDP options. The station name alias shall concatenate the Peer Port ID,”.” and Peer Chassis ID.

NOTE 4 Example for Station Name Alias = “port-001.mill-1.factory3.org”

Device Initiative
This attribute contains the value to enable or disable the issuing of the Hello service.

Attribute type: Unsigned16

Allowed values: ON, OFF

DHCP
This optional attribute contains the following suboption attributes:

Host Name
This attribute contains the value according to RFC 2132.

Attribute type: see RFC 2132

Vendor Specific Information
This attribute contains the value according to RFC 2132.

Attribute type: see RFC 2132

Server Identifier
This attribute contains the value according to RFC 2132.

Attribute type: see RFC 2132

Parameter Request List
This attribute contains the value according to RFC 2132.

Attribute type: see RFC 2132

Class Identifier
This attribute contains the value according to RFC 2132.

Attribute type: see RFC 2132

Fully Qualified Domain Name
This attribute contains the value according to RFC 2132.

Attribute type: see RFC 2132

UUID/GUID-based Clienet
This attribute contains the value according to RFC 2132.

Attribute type: see RFC 2132

Control DHCP for Address Resolution
This attribute contains control information for the usage of DHCP.

Attribute type: Unsigned8

Allowed Values: DONT_USE_DHCP, DONT_USE_AND_RESET_DHCP_OPTIONS, USE_DHCP

Manufacturer Specific
This optional attribute contains the following suboption attributes:

List of Manufacturer Specific Suboptions
This attribute list contains the following attributes:

Manufacturer Specific Suboption
This attribute contains the manufacturer specific value.

Attribute type: Unsigned8

Manufacturer OUI
This attribute contains the organizational unique identifier to identify the vendor.

Attribute type: OctetString, Length 3

Manufacturer Specific String
This attribute contains the individual information.

Attribute type: OctetString

Protocol machine Parameter
This attribute contains the following attributes to describe the behavior of the protocol machines:

Max Retry Limit
This attribute contains the maximum limit to repeat a client service in case no answer has been received by the client.

Attribute type: Unsigned8

Default Value: 4

Allowed Values: 0 – 15

UC Client Timeout
This attribute contains the maximum value in seconds, which a client waits for a unicast response.

Attribute type: Unsigned16

Default Value: 1

Allowed Values: 1 – 30

MC Client Timeout
This attribute contains the maximum value in milliseconds, which a client waits for all possible identify responses. It shall be greater then the calculated Response Delay Time

using the Response Delay Factor from the request. It shall be 400 ms if the Response Delay Factor equals one and rounded up to a full second plus 1 s otherwise.

Attribute type: Unsigned16

Allowed Values: 400 – 65 000

Tag Control Information
This attribute contains the following attributes:

VLAN ID
This attribute contains the VLAN ID according to IEEE 802.1Q.

Attribute type: Unsigned16

Allowed Value: 0

Priority
This attribute contains the priority tag of the frame according to IEEE 802.1Q.

Attribute type: Unsigned8

Allowed Value: 0

Client Hold Time
This attribute contains the maximum value in seconds, which a server shall only allow Get or Set services from the last active client. Service requests from other server shall be ignored before.

Attribute type: Unsigned16

Default Value: 3

6.3.1.3 DCP service specification

6.3.1.3.1 Get

The Get service is used to read one or more ASE attributes by means of the client server communication model. The attribute group shall be addressed by means of an option. The subgroup shall be addressed by means of a suboption.

The server shall respond with the individual status “not supported” if optional options or suboptions are not available. Generally, the requested data shall be responded as long as they fit in the DCP-Get-ResPDU. Otherwise, the status “resource error” for the first not fitting suboption shall be responded. If the requested option with suboption contains a list with more than one element then all available list elements shall be responded as long as they fit in the PDU. The element order is arbitrary.

Table 43 shows the parameter of the service.

Table 43 – Get

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
DA	M	M(=)		
List Of Options	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Result(+)			S	S(=)
List Of Data			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
Length			M	M(=)
Status			M	M(=)
IP Info			U	U(=)
Data			U	U(=)
AddUserData			U	U(=)
Result(-)			S	S(=)
ERRCLS			M	M(=)
ERRCODE			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

DA
This parameter shall contain the individual IEEE 802.3 MAC address of the server.

List Of Options
This parameter contains the requested list with options with their related suboptions. These parameters shall correspond with the ASE attributes.

Option
This parameter contains the option that shall be read from the device. It can contain the following ASE attributes: IP, Device Properties, DHCP, and Manufacturer Specific. The Control option shall not be used within this service. Furthermore, the All Selector option can be requested as an proxy option to get all available information from the server. In this case, the response shall at least contain the option IP with the suboption IP Parameter and the option Device Parameter with the suboptions Device ID, Device Role, Device Options, and Name Of Station. The response may contain further options/suboptions.

Suboption
This parameter contains the suboption corresponding with the option that shall be read from the device. Depending on the option used, only designated suboptions, which related to the grouped ASE attributes, shall be used.

Result(+)
This parameter indicates that the service request succeeded. It shall contain the requested values or an error code if an option/suboption is not available.

List Of Data
This parameter shall contain all values of the requested options with suboptions.

Option
This parameter contains the option requested from the device.

Suboption
This parameter contains the suboption requested from the device.

Length
This parameter contains the number of octets of the parameter Data and optional AddUserData depending of the parameter Status of the service response.

Status
This parameter contains the status of the parameter Data.

Allowed values: NO_ERROR– requested data fully delivered,
OPTION_NOT_SUPPORTED– requested optional Option is not supported,
SUBOPTION_NOT_SUPPORTED–requested optional Suboption is not suported,
RESSOURCE_ERROR-requested available data does not fully fit in the DCP-Get-ResPDU

NOTE The status RESSOURCE_ERROR indicates that the value of an option could not be delivered completely in case it exceeds the available size of the PDU. However, the data should be present as long as it makes sense from the application point of view. For example, if the elements of the parameter List of Alias Names as a whole exceed the size of the PDU then, however, those elements which already fit should be present to deliver the requested information at least partly.

IP Info
This optional parameter contains the additional info in case of suboption IP Parameter and the parameter Status contains the value NO_ERROR.

Allowed values: IP_VIA_SET – IP Parameter via Set service received,
IP_VIA_DHCP – IP Parameter via DHCP service received,
IP_NOT_ACTIVE – IP Parameter not active, address conflict detected

Data
This parameter contains the value of the requested ASE attribute as long as it fits completely in the DCP-Get-ResPDU. This parameter shall only be present if the Status contains the value NO_ERROR.

AddUserData
This optional parameter can contain vendor specific data in a vendor specific format. This optional response parameter may only be present if the Status contains a value other then NO_ERROR.

Result(-)
This parameter indicates that the service request failed in general for local reasons.

ERRCLS
The parameter ERRCLS contains the error class of the specific error.

Type: Unsigned16

Allowed Values: CTXT – local context invalid, PROTOCOL – transmission error

ERRCODE
The parameter ERRCODE contains the error code of the specific error.

Type: Unsigned16

Allowed Values: INVALID_STATE if ERRCLS=CTX, LMPM if ERRCLS=PROTOCOL (local error during transmission), TIMEOUT if ERRCLS=PROTOCOL (remote station was not responding)

6.3.1.3.2 Set

The Set service is used to write one or more ASE attributes by means of the client server communication model. The attribute group shall be addressed by means of an option. The subgroup shall be addressed by means of a suboption. Furthermore, special control commands control the behavior of the server. The commands Start Transaction and Stop Transaction shall be used to group a sequence of set service and the server shall be locked for a client in this case.

The server shall respond with the individual status “not supported” if optional options or suboptions are not available. The element order is arbitrary.

Table 44 shows the parameter of the service.

Table 44 – Set

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
DA	M	M(=)		
List Of Data	U	U(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Length	M	M(=)		
Data Qualifier	M	M(=)		
Manufacturer OUI	U	U(=)		
Data	M	M(=)		
List Of Control Commands	U	U(=)		
Control	M	M(=)		
Start Transaction	S	S(=)		
Stop Transaction	S	S(=)		
Factory Reset	U	U(=)		
Signal	U	U(=)		
FlashOnce	M	M(=)		
Result(+)			S	S(=)
List Of Response			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
Length			M	M(=)
Status			M	M(=)
Add User Data			U	U(=)
Result(-)				S
ERRCLS				M
ERRCODE				M

Argument
The argument shall convey the service specific parameters of the service request.

DA
This parameter shall contain the individual IEEE 802.3 MAC address of the server.

List Of Data
This parameter contains the data to be written with options with their related suboptions. These parameters shall correspond with the ASE attributes.

Option
This parameter contains the option that shall be written to the server. It can contain the following ASE attributes: IP, Device Properties, DHCP, and Manufacturer Specific. The option IP with the suboption MAC Address shall not be used.

Suboption
This parameter contains the suboption corresponding with the option that shall be written to the server. Depending on the option used, only designated suboptions, which related to the grouped ASE attributes, shall be used.

Length
This parameter contains the number of octets of the parameter.

Data Qualifier
This parameter contains an additional info of the data that has to be written to the server. Depending on the requested option/suboption the status has different meanings.

Allowed Values for Option IP: USE_TEMPORARY_AND_CLEAR_STORED_ADDRESS, SAVE_PERMANENT

Allowed Values for other Option: USE_TEMPORARY, SAVE_PERMANENT

Manufacturer OUI
This optional parameter contains the OUI for the manufacturer specific option.

Data
This parameter contains the data that has to be written to the server.

List Of Control Commands
This parameter contains the following parameter:

Control
This parameter contains the following suboption parameter:

Start Transaction
This parameter shall mark the start of a sequence of different Set services. The server shall lock the ASE attributes for the requesting client until the Stop Transaction is indicated or a certain timeout occurs.

Attribute type: Unsigned8

Allowed Value: START

Stop Transaction
This parameter shall mark the end of a sequence of different Set services. The server shall unlock the ASE attributes.

Attribute type: Unsigned8

Allowed Value: STOP

Signal
This parameter contains the following parameter:

Flash Once
This parameter shall contain the signal value. Flash Once means to flash the Ethernet LINK LED (or an alternative signalization) with a duration of 3 s with a frequency of 1 Hz (500 ms on, 500 ms off).

Attribute type: Unsigned16

Allowed Value: FLASH_ONCE

Reset To Factory Settings
This parameter shall be used to reset all DCP attributes by means of the Set services.

Attribute type: Unsigned8

Allowed Value: RESET_TO_FACTORY_SETTINGS

Result(+)
This parameter indicates that the service request succeeded. It shall contain the requested values.

List Of Response
This parameter contains the return values of the write operation of the requested option/suboption.

Option
This parameter contains the option requested from the device.

Suboption
This parameter contains the suboption requested from the device.

Length
This parameter contains the number of octets of the parameter Data and optional AddUserData depending of the parameter Status of the service response.

Status
This parameter contains the status of the parameter Data transport. The value NO_ERROR signals the delivery and if requested the permanent storage of the data. It may take some time before the parameter is working.

Allowed values: NO_ERROR– requested data fully delivered,
OPTION_NOT_SUPPORTED– requested optional Option is not supported,
SUBOPTION_NOT_SUPPORTED–requested optional Suboption is not supported,
SET_NOT_POSSIBLE – set not possible for local reasons,
IN_OPERATION – set not possible because of operating application

AddUserData
This optional parameter can contain vendor specific data in a vendor specific format. This optional response parameter may only be present if the Status contains a value other then NO_ERROR.

Result(-)
This parameter indicates that the service request failed in general for local reasons.

ERRCLS
The parameter ERRCLS contains the error class of the specific error.

Type: Unsigned16

Allowed Values: CTXT – local context invalid, PROTOCOL – transmission error

ERRCODE
The parameter ERRCODE contains the error code of the specific error.

Type: Unsigned16

Allowed Values: INVALID_STATE if ERRCLS=CTX, LMPM if ERRCLS=PROTOCOL (local error during transmission), TIMEOUT if ERRCLS=PROTOCOL (remote station was not responding)

6.3.1.3.3 Identify

The Identify service can be used to discover devices on the network. The service shall be sent to a well known multicast IEEE 802.3 MAC address that is defined by the DCP protocol. Only devices that fulfill all requested filter shall reply to this request. The Identify confirmation shall contain all responses received within a certain time interval. The responding devices are identified by the IEEE 802.3 source MAC address. Otherwise, the confirmation shall contain an empty list.

In case the filter criteria are matched a server shall respond with the data for options and suboptions that are part of the filter. In addition, the data of option

- IP with suboption IP Parameter and its elements IP Address, Subnet Mask, and Standard Gateway,
- Device Properties with its suboptions List of Station Names, Device ID, Device Role, and Device Option

shall always be included in the response. The sequence of data blocks may be arbitrary.

Table 45 shows the parameter of the service.

Table 45 – Identify

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
List Of Filter	S	S(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Length	M	M(=)		
Data	M	M(=)		
All Selector	S	S(=)		
Response Delay Factor	M			
Result(+)			S	S(=)
List of Devices			U	U(=)
SA				M
List of Data			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
Length			M	M(=)
Data			U	U(=)
Result(-)			S	S(=)
ERRCLS			M	M(=)
ERRCODE			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

List Of Filter
This conditional parameter contains the options/suboptions that shall be used to filter the request on the server side. More than one filter shall be handelded with an AND operation at the server side.

NOTE AND operation means that a server only responds to an identify request within the response delay time if all filter criteria match with currently used attribute values.

Option
This parameter contains the value of the option that shall be used to filter the request on the server side. It can contain the following ASE attributes: IP, Device Properties, DHCP, and Manufacturer Specific.

Suboption
This parameter contains the value of the suboption that shall be used to filter the request on the server side. Depending on the option used, only designated suboptions, which related to the grouped ASE attributes, shall be used.

Length
This parameter contains the length of the data of the option and suboption that shall be used to filter the request on the server side.

Data
This parameter contains the data of the option and suboption that shall be used to filter the request on the server side.

All Selector
This conditional parameter indicates that all devices shall at least respond with their mandatory options.

Response Delay Factor
This parameter contains the factor for the server to calculate the spread value to delay the response if the List Of Filter criteria are fulfilled. The time base is 10 ms. The calculated delay of the server shall be between 10 ms and 64 s.

Parameter type: Unsigned16

Allowed Values: 1 – 6 400

Default Values: 1 (IO Controller)

Result(+)
This parameter indicates that the service request succeeded. It shall contain the requested values.

Result(-)
This parameter indicates that the service request failed in general for local reasons.

ERRCLS
The parameter ERRCLS contains the error class of the specific error.

Type: Unsigned16

Allowed Values: CTXT – local context invalid, PROTOCOL – transmission error

ERRCODE
The parameter ERRCODE contains the error code of the specific error.

Type: Unsigned16

Allowed Values: INVALID_STATE if ERRCLS=CTX, LMPM if ERRCLS=PROTOCOL (local error during transmission), TIMEOUT if ERRCLS=PROTOCOL (remote station was not responding)

6.3.1.3.4 Hello

The optional Hello service can be used to announce the presence of a device in a network. The service shall be sent to a well known multicast IEEE 802.3 MAC address that is defined by the DCP protocol.

The service shall transmit the requested number of DCP-Hello-ReqPDUs within a certain restricted time period. The requested hello interval shall be used to wait before each retransmission.

NOTE The Hello service is intended to use to accelerate the startup phase therefore transmission should not remain longer than a view hundred milliseconds.

Table 46 shows the parameter of the service.

Table 46 – Hello

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
List of Data	M	M(=)	
Name of Station	M	M(=)	
IP Parameter	M	M(=)	
IP Address	M	M(=)	
Subnet Mask	M	M(=)	
Standard Gateway	M	M(=)	
Device ID	M	M(=)	
List of Device Options	M	M(=)	
Device Option	M	M(=)	
Device Suboption	M	M(=)	
Length	M	M(=)	
Data	M	M(=)	
Device Role	M	M(=)	
Number of DCP-Hello-ReqPDUs	M		
Initial Delay	M		
Hello Interval	M		
Result(+)			S(=)
Result(-)			S(=)

Argument
The argument shall convey the service specific parameters of the service request.

List Of Data
This parameter list contains of the following parameter.

Name Of Station
This parameter contains the value of the corresponding ASE attribute.

IP Parameter
This parameter consists of the following parameter:

IP Address
This parameter contains the value of the corresponding ASE attribute.

Subnet Mask
This parameter contains the value of the corresponding ASE attribute.

Standard Gateway
This parameter contains the value of the corresponding ASE attribute.

Device ID
This parameter contains the value of the corresponding ASE attribute.

List of Device Options
This parameter list consists of the following parameter:

Device Option
This parameter contains the value of the corresponding ASE attribute.

Device Suboption
This parameter contains the value of the corresponding ASE attribute.

Length
This parameter contains the length of the data.

Data
This parameter contains the data of the option and suboption.

Device Role
This parameter contains the value of the corresponding ASE attribute.

Device Initiative
This parameter contains the value of the corresponding ASE attribute.

Number of DCP-Hello-ReqPDUs
This parameter contains the number of DCP-Hello-ReqPDUs that shall be issued with the service request.

Parameter type: Unsigned32

Allowed Values: 1-15

Default Values: 3

Hello Interval
This parameter contains the time interval between transmitting the first and each subsequent DCP-Hello-ReqPDU.

Parameter type: Unsigned16

Allowed Values: 30_MILLISECONDS, 50_MILLISECONDS, 100_MILLISECONDS, 300_MILLISECONDS, 500_MILLISECONDS, 1_SECONDS

Default Values: 30_MILLISECONDS

Result(+)
This parameter indicates that the service request succeeded. It shall contain the requested values.

Result(-)
This parameter indicates that the service request failed in general for local reasons.

6.3.2 Precision time control ASE

6.3.2.1 Concepts

The "Precision Transparent Clock Protocol" (PTCP) represents a uniform method of distributing the clock on several stations. In particular, this means that a PTCP master synchronizes an amount of PTCP slaves. It also supports a PTCP best master model and up to 32 different clocks. The transport protocol PTCP is data link layer based.

The following application scenario shows how PTCP can be used to synchronize the time and cyclic communication in the sub-microsecond-range. To achieve the required synchronization, special support is required, see Figure 3.

- Measurement of the line delay between two corresponding ports
- Clock synchronization between PTCP master and PTCP slaves

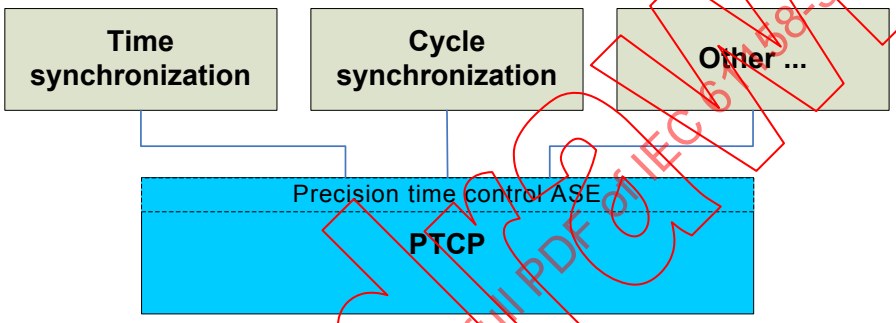


Figure 3 – PTCP applications

6.3.2.2 PTCP class specification

6.3.2.2.1 Template

ASE: **PTCP ASE**
CLASS: **PTCP**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Sync ID
2	(m)	Attribute:	Domain
2.1	(m)	Attribute:	Domain UUID
2.2	(m)	Attribute:	Sequence ID
2.3	(m)	Attribute:	Sync Send Factor
2.4	(m)	Attribute:	PTCP Takeover Timeout
2.5	(m)	Attribute:	PTCP Timeout
2.6	(m)	Attribute:	Role (MASTER, SLAVE)
2.7	(c)	Constraint:	Role = MASTER
2.7.1	(m)	Attribute:	Priority
2.7.2	(m)	Attribute:	Accuracy
2.7.3	(m)	Attribute:	Variance
2.7.4	(m)	Attribute:	PTCP Master Startup Time
2.8	(c)	Constraint:	Sync ID = TIME
2.8.1	(m)	Attribute:	PTCP Time
2.8.1.1	(m)	Attribute:	Epoch Number
2.8.1.2	(m)	Attribute:	Seconds
2.8.1.3	(m)	Attribute:	Nano Seconds

2.8.1.4	(m)	Attribute:	Current UTC Offset
2.9	(c)	Constraint:	Sync ID = CLOCK
2.9.1	(m)	Attribute:	Local Time
2.10	(m)	Attribute:	PLL Window
2.11	(m)	Attribute:	Sync State Info
2.11.1	(m)	Attribute:	Jitter out of Boundary
2.11.2	(m)	Attribute:	No Sync Message Received
2.12	(m)	Attribute:	Rate Compensation Factor
2.13	(m)	Attribute:	List of Ports
2.13.1	(m)	Attribute:	PortID
2.13.2	(m)	Attribute:	Borderline
2.13.2.1	(m)	Attribute:	Ingress
2.13.2.2	(m)	Attribute:	Egress
2.13.3	(m)	Attribute:	Line Delay
2.13.4	(m)	Attribute:	Rate Compensation Factor Peer
2.14	(o)	Attribute:	List of OUIs
2.14.1	(m)	Attribute:	OUI
2.14.2	(s)	Attribute:	Subtype
2.14.3	(s)	Attribute:	Data Block

SERVICES:

1	(m)	OpsService:	Start Bridge
2	(m)	OpsService:	Start Slave
3	(o)	OpsService:	Start Master
4	(m)	OpsService:	Stop Bridge
5	(m)	OpsService:	Stop Slave
6	(o)	OpsService:	Stop Master
7	(m)	OpsService:	Sync State Change

6.3.2.2.2 Attributes

Sync ID
This key attribute identifies the instance of the protocol machine to which the data belong. The value CLOCK shall be used for cycle synchronization. The value TIME shall be used for time synchronization.

NOTE Cycle synchronization and time synchronization are concurrently used if a device has both modes activated.

Attribute type: Unsigned8

Allowed values: CLOCK, TIME

Domain
This attribute consists of the following elements:

Domain UUID
This attribute contains the UUID of the domain for synchronization which is provided by project planning and issued by a master. The UUID is used for a logical grouping of PTCP clocks that synchronize each other using the PTCP protocol. Clocks in another PTCP domains shall not be synchronized.

Attribute type: UUID

Sequence ID
This attribute contains the sequence number of the sync message. It shall be incremented with every new sync messages. By means of the Sequence ID the receiver detects duplications.

Attribute type: Unsigned16

Sync Send Factor
This attribute contains the send interval for sync messages.

Attribute type: Unsigned16

Allowed values: 0 to 64 000

PTCP Takeover Timeout
This attribute contains a timeout value for detection the loss of sync messages from the current sync master. The sync slave shall try to find a new sync master. The PTCP takeover timeout value shall be in the range from 32 ms to 16 352 ms for Sync ID = CLOCK and in the range from 32 ms to 2 759 400 000 ms for Sync ID = TIME.

Attribute type: Unsigned32

Default value: 96 ms for Sync ID = CLOCK, 9 600 ms for Sync ID = TIME

PTCP Timeout
This attribute contains a timeout value for detection the loss of any sync messages. The PTCP timeout value shall be in the range from 32 ms to 16 352 ms for Sync ID = CLOCK and in the range from 32 ms to 2 759 400 000 ms for Sync ID = TIME.

Attribute type: Unsigned32

Default value: 192 ms for Sync ID = CLOCK, 19 200 ms for Sync ID = TIME

Role
This attribute contains the role to time synchronization.

Attribute type: Unsigned16

Allowed values: MASTER, SLAVE

Priority
This attribute shall be the priority of master issuing sync messages.

Attribute type: Unsigned8

Allowed values: PRIMARY, SECONDARY

Accuracy
This attribute shall be the accuracy of master time in comparison with “Coordinated Universal Time” / “International Atomic Time” for Sync ID=TIME and the precision of the local clock for Sync ID=CLOCK.

Attribute type: Unsigned8

Allowed values: 25_ns, 100_ns, 250_ns, 1_µs, 2,5_µs, 10_µs, 25_µs, 100_µs, 250_µs, 1_ms, UNKNOWN

Default value: 100_ns for Sync ID = CLOCK and Sync ID = TIME

Variance
This attribute shall be the value for the quality of the clock. Each master clock shall maintain an estimation of its inherit precision. The PTCP variance is based on the theory of Allan deviation. The Allan deviation is $\sigma_y(\tau)$ is estimtated as follows:

$$\sigma_y(\tau)=\left[\frac{1}{2(N-2)\tau^2}\times\sum_{k=1}^{N-2}(x_{k+2}-2x_{k+1}+x_k)^2\right]^{\frac{1}{2}} \quad (14)$$

The PTCP variance is defined by:

$$\sigma_{PTCP}^2=\tau^2\times\frac{1}{3}\sigma_y^2 \quad (15)$$

An unbiased estimate of the PTCP variance shall be computed as follows:

$$\sigma_{PTCP}^2=\frac{1}{3}\left[\frac{1}{2(N-2)}\times\sum_{k=1}^{N-2}(x_{k+2}-2x_{k+1}+x_k)^2\right] \quad (16)$$

x_k, x_{k+1} and x_{k+2} are time residual measurement, made at times t_k, $t_k + \tau$ and $t_k + 2\tau$, between the time provided by the measured clock and a local reference clock. N is the

number of data samples. For PTCP variance the quantity τ, the sample period, shall be the value defined as sync interval.

PTCP variances shall be represented as followos:

An estimate of the variance, is computed in units of seconds squared.

The logarithm to the base 2 of this estimate is computed. The computation of the logarithm need not be more precise than the precision of the estimate of the variance.

The logarithm is multiplied by 28 to produce a scaled value.

The value is represented as a 2's complement Integer16. The value 0x8000 is added to the reported value represented in this form and any overflow is ignored. The result, i.e., the offset scaled reported value, is cast as an Unsigned16.

This offset scaled value, represented as Unsigned16, shall be the value of the log variances.

The largest possible positive number, 0xFFFF, for the offset scaled log variance attribute shall indicate that the variance is either too large to be represented, or has not been computed. The variance values are used in the selection of the best master clock.

Attribute type: Unsigned16

Note 1 For example, suppose the PTCP variance value is $1\ 414x2^{-73} = 1\ 497x10^{-22}\ s^2$. Then, $\log_2(1\ 414x2^{-73}) = -73 + 0.5 = -72.5$. If this were expressed as an Integer16, it would truncate to -72. To retain some precision the value is scaled by 2^8 to yield a scaledLogVariance of -18 560, 0xB780, which retains 8 bits more precision. To this is added 0x8000 to yield the offset scaled reported value 0x3780.

PTCP Master Startup Time
This attribute shall be the duration of the startup phase of the sync master. During this time the sync master shall check if there is a sync master with a higher clock quality in the sync domain.

Attribute type: Unsigned16

Allowed values: 0 to 300 s

Default value: 10 s

Time
This attribute consists of the following elements for the time representation:

PTCP Time
This attribute consists of the following elements for the PTCP time representation since the begin of the PTCP epoch 1970-01-01 T 00:00:00:

Epoch Number
This attribute shall be the current number of times the 32-bit Seconds clock has rolled over since the begin of the PTCP epoch.

Attribute type: Unsigned16

Seconds
This attribute contains the current number of seconds since 1970-01-01 T 00:00:00.

Attribute type: Unsigned32

Nano Seconds
This attribute contains the current number of nanoseconds until the next second.

Attribute type: Unsigned32

Allowed values: 0 to 999 999 999

Current UTC Offset
This attribute contains the current number of leap seconds as offset from the PTCP Time to the Coordinated Universal Time (UTC).

Attribute type: Interger16

Local Time
This attribute consists of a local time value in nanoseconds. This value shall be used for PTCP clock synchroniztion and shall be the basis for the cycle counter.

Attribute type: Unsigned64

PLL Window
This attribute shall characterize the synchronization accuracy. The deviation between the local time and the reference time issued by a master shall not exceed the value of the PLL Window.

Attribute type: Unsigned32

Sync State Info
This element consists of the following error attributes. The synchronization achieved the state synchronous if there is no error.

Jitter out of Boundary
This attribute identifies whether the necessary synchronization accurancy is achieved or not.

Attribute type: Boolean

No Sync Message Received
This attribute identifies whether synchronization messages have been received or not.

Attribute type: Boolean

Wrong PTCP Domain UUID
If an synchronization messages with wrong Domain UUID have been received, this attribute will be set.

Attribute type: Boolean

Rate Compensation Factor
This attribute shall be used to adjust the bridging time from sync and delay messages.

Attribute type: Unsigned32

List of Ports
This attribute consists of the following elements:

PortID
This attribute identifies one Port of a bridge.

Attribute type: Unsigned8

Borderline
Both subsequent attributes shall contain the value ON for end-ports of a PTCP Domain. The value of the Ingress attribute shall be ON and the value for the Egress atrribute shall be OFF for an port coupling two PTCP segments, for the hierarchical higher segment. The value of the Ingress attribute shall be OFF and the value for the Egress atrribute shall be ON for an port coupling two PTCP segments, for the hierarchical lower segment.

NOTE 2 It is intended to avoid transmission from PTCP sync and PTCP announce messages from a hirarchical lower segment to a higher segment within a PTCP domain

This attribute consists of the following elements:

Ingress
This attribute identifies the end-ports of a PTPC Domain.

Attribute type: Unsigned8

Allowed values: ON, OFF

Egress
This attribute identifies the end-ports of a PTPC Domain or a level / segment borderline.

Attribute type: Unsigned8

Allowed values: ON, OFF

Line Delay
This attribute contains the transmit latency from this port to the connected port.

Attribute type: Unsigned32

Rate Compensation Factor Peer
This attribute contains the ratio between local quarz frequency and the remote quarz frequency of the device at the connected port.

Attribute type: Float64

List of OUIs
This attribute consists of the following elements:

OUI
This attribute is an Organizationally Unique Identifier as defined by IANA.

Attribute type: Octet[3]

Subtype
This attribute is an organizationally specific parameter valid in the context of the attribute OUI.

Attribute type: Unsigned8

Data Block
This attribute is an organizationally specific parameter valid in the context of the attributes OUI and Subtype.

Attribute type: UUID

6.3.2.3 PTCP service specification

6.3.2.3.1 Start bridge

The start bridge service creates a instance of the PTCP protocol machine. Table 47 shows the parameter of the service.

Table 47 – Start bridge

Parameter name	Req	Cnf
Argument	M	
Sync ID	M	
List ofPort Parameter	U	
Port ID	M	
Borderline	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

List of Port Parameter
This optional list of parameter consist of the following parameter:

Port ID
This parameter contains the ID of the port which issues the delay request message

Borderline
This parameter contains the end-ports of a clock-domain

Result(+)
This parameter indicates that the service request succeeded.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter Error code contains the error code of the specific error.

Type: Unsigned16

Allowed Values: ENTRY_NOT_POSSIBLE, SYNC_ID_EXISTS

6.3.2.3.2 Start slave

The start slave service adds synchronization properties to the PTCP protocol machine with. Table 48 shows the parameter of the service.

Table 48 – Start slave

Parameter name	Req	Cnf
Argument	M	
SyncID	M	
Subdomain UUID	M	
Sync Send Factor	M	
PTCP Timeout Factor	M	
PLL Window	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Subdomain UUID
This parameter contains the UUID of the synchronization subdomain. The UUID is used for a logical grouping of PTCP clocks that synchronize each other using the PTCP protocol.

Sync Send Factor
This parameter contains the reception interval of sync messages.

PTCP Timeout Factor
This parameter contains a timeout value to detect the lost of sync messages.

PLL Window
This parameter contains the required synchronization accuracy. The deviation between the local time and the reference time issued by a grandmaster shall not exceed the value of the PLLWindow.

Result(+)
This parameter indicates that the service request succeeded.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter Error code contains the error code of the specific error.

Type: Unsigned16

Allowed Values: SLAVE_NOT_POSSIBLE, SLAVE_EXISTS

6.3.2.3.3 Start master

The start master service adds synchronization properties with the qualification to become synchronization master to the PTCP protocol machine. Table 49 shows the parameter of the service.

Table 49 – Start master

Parameter name	Req	Cnf
Argument	M	
Sync ID	M	
Subdomain UUID	M	
Sync Send Factor	M	
Sync Class	M	
Clock Stratum	M	
Current UTC Offset	M	
Epoch Number	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Subdomain UUID
This parameter contains the UUID of the synchronization subdomain. The UUID is used for a logical grouping of PTCP clocks that synchronize each other using the PTCP protocol.

Sync Send Factor
This parameter contains the send interval of sync messages.

Sync Class
This parameter contains the recommended role (primary, secondary) as a master.

Clock Stratum
This parameter contains the quality of the own time.

Current UTC Offset
This parameter contains the current offset to the Coordinated Universal Time (UTC).

Epoch Number
This parameter contains the current number of times the 32-bit Seconds clock has rolled over since the begin of the PTCP epoch. The PTCP epoch began at 0 hours on 1. January 1970.

Result(+)
This parameter indicates that the service request succeeded.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter Error code contains the error code of the specific error.

Type: Unsigned16

Allowed Values: MASTER_NOT_POSSIBLE, MASTER_EXISTS, MASTER_NOT_EXISTS

6.3.2.3.4 Stop bridge

This service shall be used to stop the PTCP protocol machine. Table 50 shows the parameter of the service.

Table 50 – Stop bridge

Parameter name	Req	Cnf
Argument	M	
Sync ID	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(+)
This parameter indicates that the service request succeeded.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter ErrCode contains the error code of the specific error.

Type: Unsigned16

Allowed Values: SYNC_ID_NOT_EXISTS

6.3.2.3.5 Stop slave

This service shall be used to reduce the properties of the PTCP protocol machine. Synchronization will be stopped and bridge functionality remains. Table 51 shows the parameter of the service.

Table 51 –Stop slave

Parameter name	Req	Cnf
Argument	M	
Sync ID	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(+)
This parameter indicates that the service request succeeded.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

SyncID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter Error code contains the error code of the specific error.

Type: Unsigned16

Allowed Values: SLAVE_NOT_EXISTS

6.3.2.3.6 Stop master

This service shall be used to reduce the properties of the PTCP protocol machine. Master functionality will be removed and the protocol machine is forced to take over slave role. Table 52 shows the parameter of the service.

Table 52 – Stop master

Parameter name	Req	Cnf
Argument	M	
Sync ID	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(+)
This parameter indicates that the service request succeeded.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter Error code contains the error code of the specific error.

Type: Unsigned16

Allowed Values: MASTER_NOT_EXISTS

6.3.2.3.7 Sync state change

This service shall be used to indicate a change of the synchronization state. A Change can only appear for a synchronization slave. Table 53 shows the parameter of the service.

Table 53 – Sync state change

Parameter name	Ind
Argument	M
Sync ID	M
Sync State	M
Error TypeList	C

Argument
The argument shall convey the service specific parameters of the service request.

Sync ID
This ist the key attribute to identify the instance of the protocol machine.

Sync State
This ist the attribute shall contain the value SYNCHRONIZED or NOT_SYNCHRONIZED.

Error TypeList
This attribute consists of the following elements:

ErrorType
This conditional attribute identifies a synchronization error in case of Sync State=NOT_SYNCHRONIZED.

Attribute type: Unsigned16

Allowed Values:JITTER_OUT_OF_BOUNDARY, NO_SYNC_MESSAGE_RECEIVED, WRONG_PTCP_DOMAIN

Appear
This attribute identifies whether the error appears or disappears.

Attribute type: Boolean

Allowed Values: TRUE, FALSE

6.3.2.3.8 PTCP behavior

6.3.2.3.8.1 Synchronization

The synchronization is done with a sync- and an optional follow up-frame. The follow up-frame is necessary, if the hardware isn't able to add bridge and line delays on the fly.

The clock issuing the sync frames in a PTCP subdomain is called PTCP master.

Clocks which receive sync frames and adjust their time to the PTCP master are called PTCP slaves. Their synchronization accuracy depends on the accuracy of the time stamping units from each bridge which measures the line and bridge delays.

A bridge with the following enhancements is called a PTCP transparent clock:

- Line-Delay measurement
- Bridge Delay measurement
- Rate compensation for line and bridge delays
- Support media redundancy mechanism for PTCP

6.3.2.3.8.2 Rate control compensation

The rate control compensation equalizes the clock rate of a PTCP transparent clock to the PTCP master. Due to this the accumulated bridge and line delays, received by the PTSP slave to achieve higher synchronization accuracy, are measured with the clock rate of the PTCP master.

A PTCP transparent clock use sync frames to measure the clock rate relating to the PTCP master.

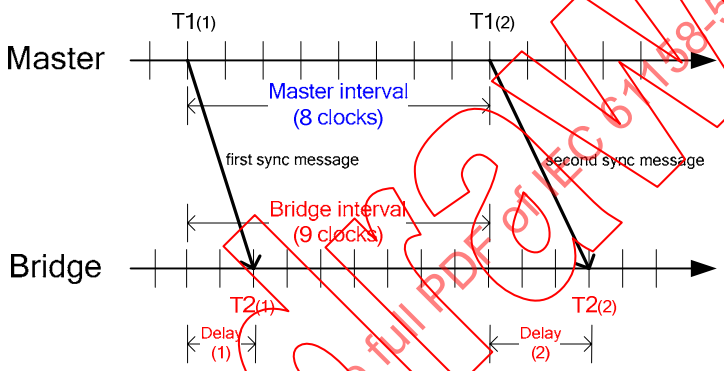


Figure 4 – Clock drift measurement

RATECOMPFACTOR = [T1(2) – T1(1)] / [(T2(2) – DELAY(2)) – (T2(1) – DELAY(1))] **(17)**

The PTCP transparent clock use the rate compensation factor to correct measured residential times.

COMPBRIDGEDELAY = RATECOMPFACTOR * BRIDGEDELAY **(18)**

6.3.2.3.8.3 Multiple synchronization

PTCP shall be used for clock, time and other kind of synchronization. With PTCP 32 kinds of synchronization are possible. There can be various sources to which all other clocks synchronize. The subdomains are not synchronized at PTCP. Each PTCP slave shall synchronize itself. The delay measurement shall be done for each communication path. The support of clock and time synchronization is are mandatory.

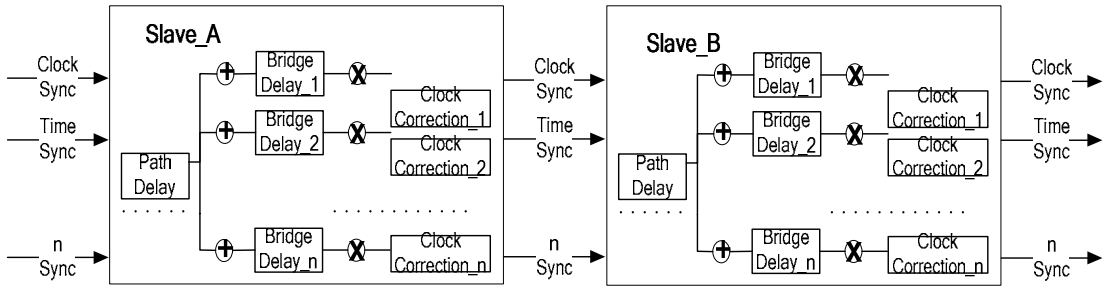


Figure 5 – Multiple synchronization

The Clock Correction consists of frequency offset and drift error of the oscillators. Each Transparent Clock shall correct the residence time of the sync and delay messages with the corresponding value of Clock Correction.

6.3.2.3.8.4 Best master algorithm (BMA)

The Best Master Algorithm specifies an algorithm how a clock determines the preferred clock issuing sync messages. The algorithm runs independently on each clock (master/slave) in a PTCP system. Each clock makes its own decision for the best master.

In computing the best master a clock shall:

- Consider the quality of the most recent sync messages received.
- Include the results of the previous best master computing. However if there are more recent qualified sync messages, the values from that messages shall be considered.

For analyzing the quality of sync message a clock shall consider:

- The master issuing the sync message belongs to the own subdomain.
- Receiving the sync messages periodically.
- The stratum of the master clock.
- The variance which describes the characteristic of the master clock.

If stratum and variance of different masters are equal the BMA use the Source-MAC-Address to decide for the preferred clock (smallest one wins).

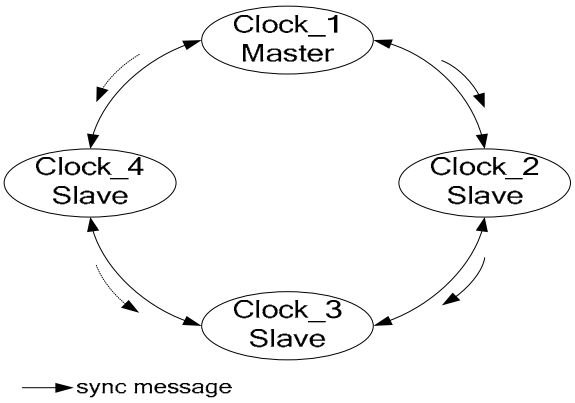
6.3.2.4 Media Redundancy support for synchronization with PTCP

To improve the reliability of a peer-to-peer network redundant communication paths are necessary. Redundant communication paths can lead to circulating frames. To avoid circulating frames the redundant communication paths are not used for data messages.

PTCP sync messages use the redundant communication paths to be independent from the reconfiguration of redundant networks and for short communication paths.

The following forwarding algorithm shall be used to avoid circulating sync messages in a network with redundant communication paths. The forwarding algorithm for PTCP takes care that on each communication path only one sync messages will be transmitted.

The first received sync message is called primary Sync message. For synchronization shall be used the primary Sync message.



6.3.2.5 Invocation of the PTCP object

For the invocation of the PTCP object the following rules apply:

– PTCP objects shall exist for each Sync ID.

6.3.3 Media redundancy ASE

6.3.3.1 Overview

The Media redundancy ASE specifies a uniform method of achieving media redundancy with deterministic recovery time in case of a single network failure (link failure, switch node failure causing link failures).

NOTE 1 Media redundancy operates transparent to the application. No changes in the application are required.

The Media Redundancy Protocol (MRP) is designed to react deterministic on a single failure of a trunk link or node in the network.

MRP is based on functions of ISO/IEC 8802-3 (IEEE 802.3) and IEEE 802.1D and is located between the Data Link Layer and Application Layer (see Figure 6).

NOTE 2 Layering assumed according IEC 61158-1.

A compliant network shall have a ring topology with multiple nodes.

One of the nodes has the role of a media redundancy manager (MRM). The function of the MRM is to observe and to control the ring topology in order to react on network faults. MRM does this by sending frames on one ring port over the ring and receives them from the ring over its other ring port, and vice–versa in the other direction.

The other nodes in the ring have the role of media redundancy clients (MRC). An MRC reacts on reconfiguration frames from the MRM and can signal link changes on its ring ports.

A compliant node shall have the ability to become media redundancy manager (MRM), or media redundancy clients (MRC), or both. Each node requires an integrated switch with at least two ports (ring ports) connected to the ring.

Each node in the ring shall be able to detect the failure or recovery of a trunk link or the failure or recovery of a neighbor node.

The MRP consists of a service and a protocol entity.

The service entity specifies, in an abstract way, the externally visible service provided by the Data Link Layer in terms of:

- primitive actions and events of the service,
- parameters associated with each primitive action and event, and the form which they take, and
- interrelationship between these actions and events, and their valid sequences.

MRP defines the services provided to

- Application Layer at the boundary between the Application and Data Link layers, and
- Systems Management at the boundary between the Data Link Layer and Systems Management.

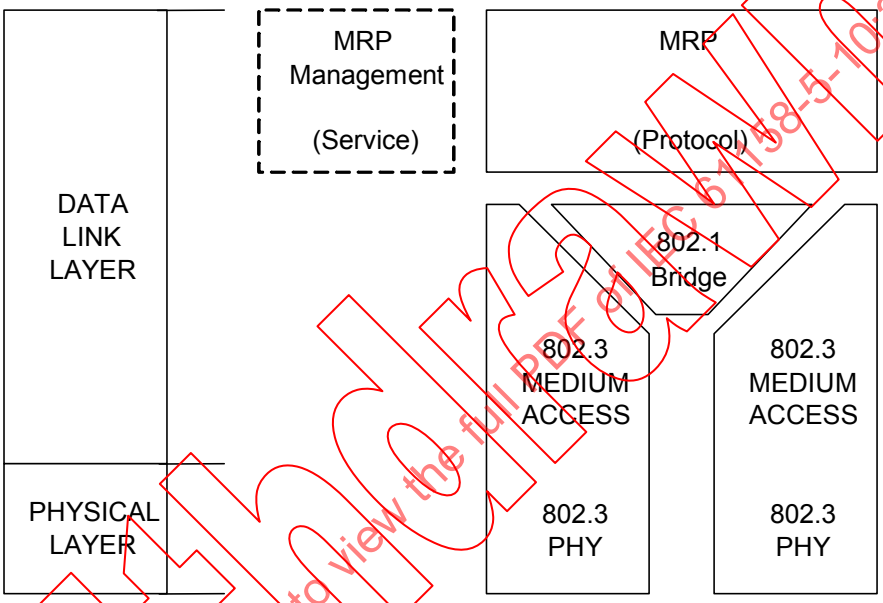


Figure 6 – MRP stack

6.3.3.2 Media redundancy class specification

6.3.3.2.1 General

The Media redundancy ASE defines one object type.

6.3.3.2.2 Template

A Media redundancy object is described by the following template:

ASE: **Media redundancy ASE**
CLASS: **Media redundancy**
CLASS ID: **not used**
PARENT CLASS: **IEEE 802.3 IEEE 802.1D IEEE 802.1 AB**
ATTRIBUTES:

1	(m)	Key Attribute:	Domain UUID
2.	(m)	Attribute:	Domain Name
3.	(m)	Attribute:	Ring Port 1 ID
4.	(m)	Attribute:	Ring Port 2 ID
5.	(o)	Attribute:	VLAN ID
6.	(o)	Attribute:	RT Redundancy (TRUE, FALSE)
7.	(m)	Attribute:	Expected Role (MANAGER, CLIENT)

8.	(c)	Constraint:	Expected Role = MANAGER
8.1	(m)	Attribute:	Manager Priority
8.2	(m)	Attribute:	Topology Change Interval
8.3	(m)	Attribute:	Topology Change Repeat Count
8.4	(m)	Attribute:	Short Test Interval
8.5	(m)	Attribute:	Default Test Interval
8.6	(m)	Attribute:	Test Monitoring Count
8.7	(c)	Constraint:	RT Redundancy = TRUE
8.7.1	(m)	Attribute:	RT Test Interval
8.7.2	(m)	Attribute:	RT Test Monitoring Count
8.8	(m)	Attribute:	Check Media Redundancy (TRUE, FALSE)
8.8.1	(c)	Constraint:	Check Media Redundancy = TRUE
8.8.1.1	(m)	Attribute:	Real Role State
8.8.1.2	(m)	Attribute:	Real Ring State
8.8.1.3	(c)	Constraint:	RT Redundancy = TRUE
8.8.1.3.1	(m)	Attribute:	Real RT Ring State
9.	(c)	Constraint:	Role = Client
9.1	(m)	Attribute:	Link Down Interval
9.2	(m)	Attribute:	Link Up Interval
9.3	(m)	Attribute:	Link Change Count
10.	(m)	Attribute:	Check Ring Port Neighborhood (TRUE, FALSE)
10.1	(c)	Constraint:	Check Ring Port Neighborhood = TRUE
10.1.1	(m)	Attribute:	Ring Port 1 Domain State
10.1.2	(m)	Attribute:	Ring Port 2 Domain State
10.1.3	(m)	Attribute:	Ring Port 1 RT State
10.1.4	(m)	Attribute:	Ring Port 2 RT State

SERVICES:

1	(m)	OpsService:	Start MRM
2	(m)	OpsService:	Stop MRM
3	(o)	OpsService:	Redundancy State Change
4	(m)	OpsService:	Start MRC
5	(m)	OpsService:	Stop MRC
6	(o)	OpsService:	Neighborhood Change

6.3.3.2.3 Attributes

Domain UUID
This key attribute defines the redundancy domain representing the ring the Media redundancy object belongs to. It is set to default domain ID or provided as unique ID by engineering.

Attribute type: UUID

Domain Name
This attribute defines the redundancy domain representing the ring the Media redundancy object belongs to. It is set to default domain name or provided as unique ID by engineering.

Attribute type: see Chassis ID

Ring Port 1 ID
This attribute specifies one port of a bridge which is assigned as ring port 1 in the redundancy domain referenced by the value of the attribute Domain ID. One port shall only belong to a single redundancy domain.

Attribute type: Unsigned16

Ring Port 2 ID
This attribute specifies another port of a bridge different from Ring Port 1 ID which is assigned as ring port 2 in the redundancy domain referenced by the value of the attribute Domain ID.

Attribute type: Unsigned16

VLAN ID
This optional attribute may be used by the Media redundancy object and specifies its VLAN identifier in the redundancy domain.

Attribute type: Unsigned16

RT Redundancy
This optional attribute specifies whether media redundancy for RT_CLASS_1 and RT_CLASS_2 frames is enabled (TRUE) or disabled (FALSE) in the redundancy domain.

Attribute type: Boolean

Expected Role
This attribute specifies the role of the Media redundancy object in the redundancy domain referenced by the value of the attribute Domain ID.

Attribute type: Unsigned16

Allowed Values: MANAGER, CLIENT

Manager Priority
This attribute contains the priority of the MRM. A lower value indicates a higher priority, 0x0000 (highest priority) to 0xF000 (lowest priority) in increments of 0x1000.

Attribute type: Unsigned16

Topology Change Interval
This attribute contains the interval for sending topology change frames.

Attribute type: Unsigned16

Topology Change Repeat Count
This attribute contains the interval count which controls repeated transmission of topology change frames.

Attribute type: Unsigned16

Short Test Interval
This attribute contains the short interval for sending test frames on ring ports after link changes in the ring.

Attribute type: Unsigned16

Default Test Interval
This attribute contains the default interval for sending test frames on ring ports.

Attribute type: Unsigned16

Test Monitoring Count
This attribute contains the interval count for monitoring the reception of test frames.

Attribute type: Unsigned16

RT Test Interval
This attribute contains the default interval for sending RT test frames on ring ports.

Attribute type: Unsigned16

RT Test Monitoring Count
This attribute contains the interval count for monitoring the reception of RT test frames.

Attribute type: Unsigned16

Check Media Redundancy
This attribute contains whether monitoring of MRM state is enabled (TRUE) or disabled (FALSE) in the redundancy domain.

Attribute type: Boolean

Real Role State
This attribute contains the actual role of the Media redundancy object in the redundancy domain.

Attribute type: Unsigned16

Allowed Values: MANAGER, CLIENT

Real Ring State
This attribute contains the actual ring state of the Media redundancy object in the redundancy domain. The Ring State shall have one of the following values:

- OPEN: Ring is open due to link or device failure in ring
- CLOSED: Ring is closed (normal operation, no error)
- UNDEFINED: Shall be set if the attribute Real Role State contains the value CLIENT (i.e. Media redundancy object was reconfigured to client role)

Attribute type: Unsigned16

Allowed Values: OPEN, CLOSED, UNDEFINED

Real RT Ring State
This attribute contains the actual RT ring state of the Media redundancy object in the redundancy domain. RT Ring State shall have one of the following values:

- OPEN: RT redundancy is lost due to ring open state or Media redundancy object in media redundancy domain with RT Redundancy Mode disabled.
- CLOSED: RT redundancy is available in the ring (normal operation, no error)
- UNDEFINED: Shall be set if the attribute Real Role State is CLIENT (i.e. Media redundancy object was reconfigured to client)

Attribute type: Unsigned16

Allowed Values: OPEN, CLOSED, UNDEFINED

Link Down Interval
This attribute contains the interval for sending link down Link Change frames on ring ports.

Attribute type: Unsigned16

Link Up Interval
This attribute contains the interval for sending link up Link Change frames on ring ports.

Attribute type: Unsigned16

Link Change Count
This attribute contains the Link Change frame count which controls repeated transmission of Link up or link down frames.

Attribute type: Unsigned16

Check Ring Port Neighborhood
This attribute contains whether the ringport neighborhood shall be checked (TRUE) or not (FALSE).

Attribute type: Boolean

Ring Port 1 Domain State
This attribute contains the media redundancy domain neighborhood state of the first ring port. It shall have one of the following values:

- GOOD: One of the following conditions is met:

  Ring port 1 is connected to a neighbor port which belongs to the same redundancy domain. The domain is not equal to the NIL domain.

  Ring port 1 and the neighbor port both do not belong to the NIL domain. They belong to different domains, but one of the two domains is the default domain.

- BAD: One of the following conditions is met:

  Ring port 1 is connected to a port which doesn't belong to a redundancy domain.

  One of the ports belongs to the NIL domain.

  Ring port 1 belongs to a different redundancy domain. None of the ports belongs to

the default domain.

– UNDEFINED: Neighborhood information not available.

Attribute type: Unsigned16

Allowed Values: GOOD, BAD, UNDEFINED

Ring Port 2 Domain State
This attribute contains the media redundancy domain neighborhood state of the second ring port. It shall have one of the following values:

- GOOD: One of the following conditions is met:

  Ring port 2 is connected to a neighbor port which belongs to the same redundancy domain. The domain is not equal to the NIL domain.

  Ring port 2 and the neighbor port both do not belong to the NIL domain. They belong to different domains, but one of the two domains is the default domain.

- BAD: One of the following conditions is met:

  Ring port 2 is connected to a port which doesn't belong to a redundancy domain.

  One of the ports belongs to the NIL domain.

  Ring port 2 belongs to a different redundancy domain. None of the ports belongs to the default domain.

– UNDEFINED: Neighborhood information not available.

Attribute type: Unsigned16

Allowed Values: GOOD, BAD, UNDEFINED

Ring Port 1 RT State
This attribute contains the media redundancy RT neighborhood state of the first ringport. It shall have one of the following values:

- GOOD: Ring port 1 is connected to a neighbor port, MRRT port status is CONFIGURED or UP on either side.
- BAD: Ring port 1 is connected to a neighbor port, the MRRT port status of one or both ports is OFF.

– UNDEFINED: Peer information not available.

Attribute type: Unsigned16

Allowed Values: GOOD, BAD, UNDEFINED

Ring Port 2 RT State
This attribute contains the media redundancy neighborhood RT state of the second ringport. It shall have one of the following values:

- GOOD: Ring port 1 is connected to a neighbor port, MRRT port status is CONFIGURED or UP on either side.
- BAD: Ring port 1 is connected to a neighbor port, the MRRT port status of one or both ports is OFF.

– UNDEFINED: Peer information not available.

Attribute type: Unsigned16

Allowed Values: GOOD, BAD, UNDEFINED

6.3.3.3 Media redundancy service specification

6.3.3.3.1 Start MRM

The Start MRM service creates a local instance of the MRM protocol machine and – if RT Redundancy Mode is enabled – an associated local instance of the MRRT protocol machine.

Table 54 shows the parameters of the service.

Table 54 – Start MRM

Parameter name	Req	Cnf
Argument	M	
Domain UUID	M	
Ring Port 1 ID	M	
Ring Port 2 ID	M	
VLAN ID	U	
RT Redundancy Mode	M	
RT Test Interval	C	
RT Test Monitoring Count	C	
Manager Priority	U	
Topology Change Interval	U	
Topology Change Repeat Count	U	
Short Test Interval	U	
Default Test Interval	U	
Test Monitoring Count	U	
Check Media Redundancy	U	
Check Ring Port Neighborhood	U	
Result(+)		S
Domain ID		M
Result(-)		S
Domain ID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Domain UUID
This parameter identifies the instance of the protocol machine.

Ring Port 1 ID
This parameter contains the ID of the port which serves as ring port 1.

Ring Port 2 ID
This parameter contains the ID of the port which serves as ring port 2.

VLAN ID
This optional parameter contains the value for the VLAN identifier.

RT Redundancy Mode
This parameter selects whether RT redundancy mode is enabled or not.

RT Test Interval
This parameter contains the value for the RT test interval in case of RT Redundancy Mode is enabled.

RT Test Monitoring Count
This parameter contains the value for the RT test monitoring count in case of RT Redundancy Mode is enabled.

Manager Priority
This parameter contains the value for the manager priority.

Topology Change Interval
This parameter contains the value of the interval for sending topology change frames.

Topology Change Repeat Count
This parameter contains the value of the interval count which controls repeated transmission of topology change frames.

Short Test Interval
This parameter contains the value of the short interval for sending test frames on ring ports after link changes in the ring.

Default Test Interval
This parameter contains the value of the default interval for sending test frames on ring ports.

Test Monitoring Count
This parameter contains the value of the interval count for monitoring the reception of test frames.

Check Media Redundancy
This parameter selects whether monitoring of MRM state is enabled or disabled.

Check Ring Port Neighborhood
This parameter selects whether monitoring of ringport neighborhood is enabled or disabled.

Result(+)
This parameter indicates that the service request succeeded.

Domain ID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Domain UUID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter Result contains the error code of the specific error.

Type: Unsigned16

Allowed Values: DOMAIN_ID_MISMATCH, ROLE_NOT_SUPPORTED, INVALID_RINGPORT

6.3.3.3.2 Stop MRM

This service shall be used to stop a local instance of the MRM protocol machine and – if RT Redundancy Mode is enabled – an associated instance of the MRRT protocol machine. Ring port states and bridge functionality remain.

Table 55 shows the parameters of the service.

Table 55 – Stop MRM

Parameter name	Req	Cnf
Argument	M	
Domain UUID	M	
Result(+)		S
Domain UUID		M
Result(-)		S
Domain UUID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Domain UUID
This parameter identifies the instance of the protocol machine.

Result(+)
This parameter indicates that the service request succeeded.

Domain UUID
This parameter identifies the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Domain UUID
This parameter identifies the instance of the protocol machine.

Error code
The parameter Result contains the error code of the specific error.

Type: Unsigned16

Allowed Values: DOMAIN_ID_MISMATCH

6.3.3.3.3 Redundancy State Change

This service shall be used to indicate a change of the media redundancy domain state.

Table 56 shows the parameters of the service.

Table 56 – Redundancy state change

Parameter name	Ind
Argument	M
Domain UUID	M
Error type List	M

Argument
The argument shall convey the service specific parameters of the service request.

Domain UUID
This parameter identifies the instance of the protocol machine.

Error type List
This attribute consists of the following elements:

Error type
This parameter identifies a media redundancy error.

Attribute type: Unsigned16

Allowed Values: MANAGER_ROLE_FAIL, CLIENT_ROLE_FAIL, RING_OPEN, RT_REDUNDANCY_LOST, MULTIPLE_MANAGERS

Appear
This attribute identifies whether the error appears or disappears.

Attribute type: Boolean

Allowed Values: TRUE, FALSE

6.3.3.3.4 Start MRC

The Start MRC service creates an instance of the MRC protocol machine.

Table 56 shows the parameters of the service.

Table 57 – Start MRC

Parameter name	Req	Cnf
Argument	M	
Domain UUID	M	
Ring Port 1 ID	M	
Ring Port 2 ID	M	
VLAN ID	U	
Link Down Interval	U	
Link Up Interval	U	
Link Change Count	U	
Check Ring Port Neighborhood	U	
Result(+)		S
Domain UUID		M
Result(-)		S
Domain UUID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Domain UUID
This parameter identifies the instance of the protocol machine.

Ring Port 1 ID
This parameter contains the ID of the port which serves as ring port 1.

Ring Port 2 ID
This parameter contains the ID of the port which serves as ring port 2.

VLAN ID
This optional parameter contains the value for the VLAN identifier.

Link Down Interval
This parameter contains the value of the interval for sending link down Link Change frames on ring ports.

Link Up Interval
This parameter contains the value of the interval for sending link up Link Change frames on ring ports.

Link Change Count
This parameter contains the value of the Link Change frame count which controls repeated transmission of Link up or link down frames.

Check Ring Port Neighborhood
This parameter whether monitoring of ringport neighborhood is enabled or disabled.

Result(+)
This parameter indicates that the service request succeeded.

Domain UUID
This ist the key attribute to identify the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Domain UUID
This ist the key attribute to identify the instance of the protocol machine.

Error code
The parameter Result contains the error code of the specific error.

Type: Unsigned16

Allowed Values: DOMAIN_ID_MISMATCH, ROLE_NOT_SUPPORTED, INVALID_RINGPORT

6.3.3.3.5 Stop MRC

This service shall be used to stop an instance of the MRC protocol machine. Ring port states and bridge functionality remain. Table 56 shows the parameters of the service.

Table 58 – Stop MRC

Parameter name	Req	Cnf
Argument	M	
Domain UUID	M	
Result(+)		S
Domain UUID		M
Result(-)		S
Domain UUID		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

Domain UUID
This parameter identifies the instance of the protocol machine.

Result(+)
This parameter indicates that the service request succeeded.

Domain UUID
This arameter identifies the instance of the protocol machine.

Result(-)
This parameter indicates that the service request failed.

Domain UUID
This arameter identifies the instance of the protocol machine.

Error code
The parameter Result contains the error code of the specific error.

Type: Unsigned16

Allowed Values: DOMAIN_ID_MISMATCH

6.3.3.3.6 Neighborhood changed

This service shall be used to indicate a change of the ring port neighborhood. Table 56 shows the parameters of the service.

Table 59 – Neighborhood changed

Parameter name	Ind
Argument	M
Domain UUID	M
Error type List	M

Argument
The argument shall convey the service specific parameters of the service request.

Domain UUID
This ist the key attribute to identify the instance of the protocol machine.

Error type List
This attribute consists of the following elements:

Error type
This element identifies a media redundancy error related to port neighbourhood.

Attribute type: Unsigned16

Allowed Values: PEER_DOMAIN_ID_MISMATCH

Port ID
This element identifies the local ringport of the Media redundancy domain which caused the error condition.

Attribute type: Unsigned16

Appear
This element identifies whether the error appears or disappears.

Attribute type: Boolean

Allowed Values: TRUE, FALSE

6.3.3.4 Media redundancy behavior

6.3.3.4.1 Ring ports

The Media Redundancy Manager MRM and the Media Redundancy Client MRC shall have two ring ports.

The MRM and MRC shall be able to detect the failure or recovery of a link on a ring port with mechanisms based on IEEE 802.3.

The MRM and MRC shall not forward MRP test frames, MRP topology change frames, and MRP link change frames to non ring ports.

A ring port shall take one of the following port states:

- DISABLED:
  All frames shall be dropped.
- BLOCKED:
  All frames shall be dropped except
  - MRP topology change frames and MRP test frames from the MRM
  - MRP link change frames from a MRC
  - frames from other protocols that also define to pass BLOCKED ports (e.g. LLDP, PTP)
- FORWARDING:
  All frames shall be passed through according to the forwarding behaviour of IEEE 802.1D.

6.3.3.4.2 Media redundancy manager (MRM)

A first ring port of the MRM shall be connected to the ring ports of the MRC. The ring port of the MRC shall be connected to the ring port of another MRC or to the second ring port of the MRM, therefore forming a ring as shown in Figure 7.

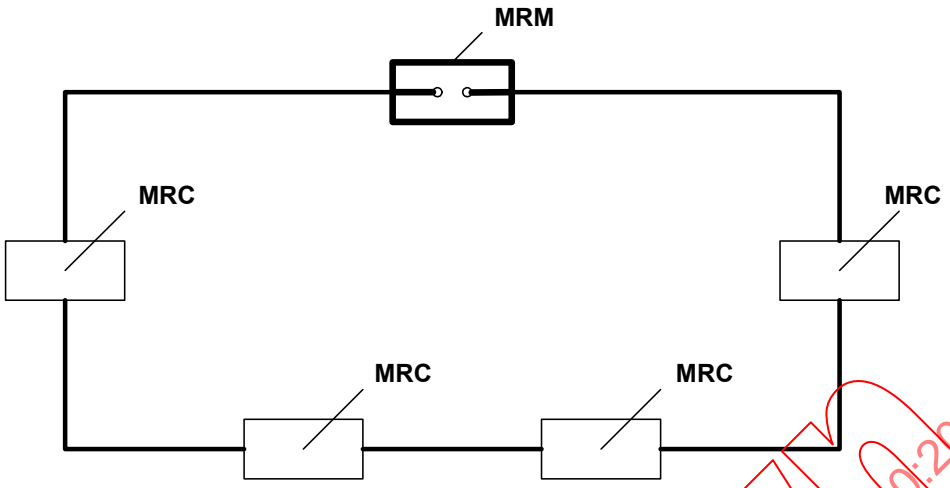


Figure 7 – Ring topology with one manager and clients

The MRM shall control the ring state:

- The MRM shall periodically send MRP test frames in a determined time interval in both directions of the ring.
- If the MRM receives its own MRP test frames (the ring is closed, see Figure 7) then it shall set one ring port in FORWARDING state and the other ring port in BLOCKED state.
- If the MRM does not receive its own MRP test frames in a determined time according to Table 61 (the ring is open, see Figure 8), then it shall set both ring ports in FORWARDING state (see Figure 8).

NOTE In order to support alarm transmission with default setup (retransmit time interval 100 ms, retransmit count 2), a reconfiguration time of < 200 ms is required.

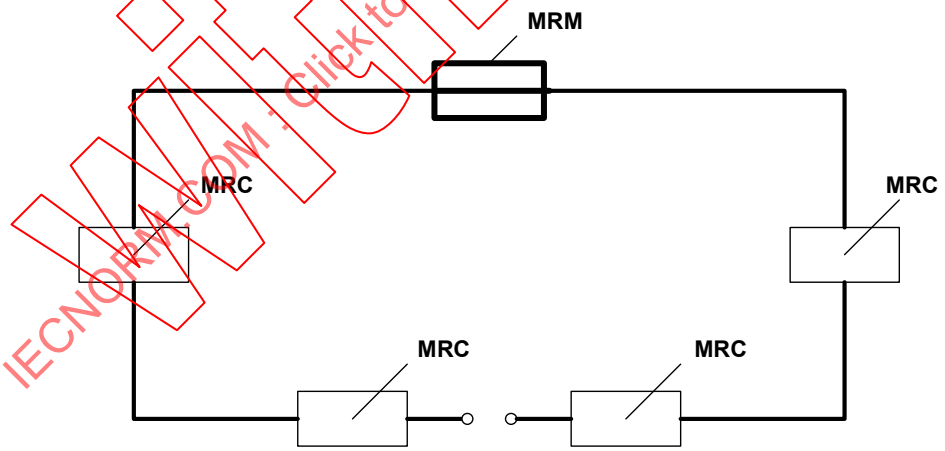


Figure 8 – MRM in an open ring

The MRM shall indicate changes in the ring state to the MRC by means of MRP topology change frames.

The MRM shall not forward MRP specific frames (MRP test frames, MRP topology change frames, MRP link change frames) between ring ports.

6.3.3.4.3 Media redundancy client (MRC)

Each MRC shall forward MRP test frames received on one ring port to the other ring port and vice versa.

If the MRC detects a failure or recovery of a ring port link, the MRC may optionally notify the change by sending MRP link change frames through both of its ring ports.

Each MRC shall forward MRP link change frames received on one ring port to the other ring port and vice versa.

Each MRC shall forward MRP topology change frames received on on one ring port to the other ring port and vice versa. Each MRC shall process these frames. It shall clear its filtering data base (FDB) if requested by a MRP topology change frame in a given time interval (see Table 61, MRP_TOPchgT). The time interval is used to synchronize switchover in the ring.

6.3.3.4.4 Redundancy domain

The redundancy domain serves as key attribute representing a ring. By default, all MRM and MRC belong to the default domain. Engineering may provide a unique domain ID, especially if a MRM or MRC is member in multiple rings. A node shall assign exactly two unique ring ports per redundancy domain. A ring port shall not belong to multiple redundancy domains.

6.3.3.4.5 Multiple MRM in a single ring (option)

There shall be only one active MRM in the ring even if several nodes have this ability. Otherwise the ring will be divided into several segments.

As an option, in case of more than one node have the ability of becoming MRM in the ring, an enhanced protocol not specified in this document may be used to decide which one of these nodes shall become the MRM, while the other nodes take over the MRC role as shown in Figure 9. To this effect the nodes with the MRM ability have different priorities that shall be conveyed in the MRP_Prio field of the MRP test frame.

If an optional protocol for multiple MRM in a single ring is used then all MRM in the ring shall support the same protocol. The vendor shall specify the supported protocols.

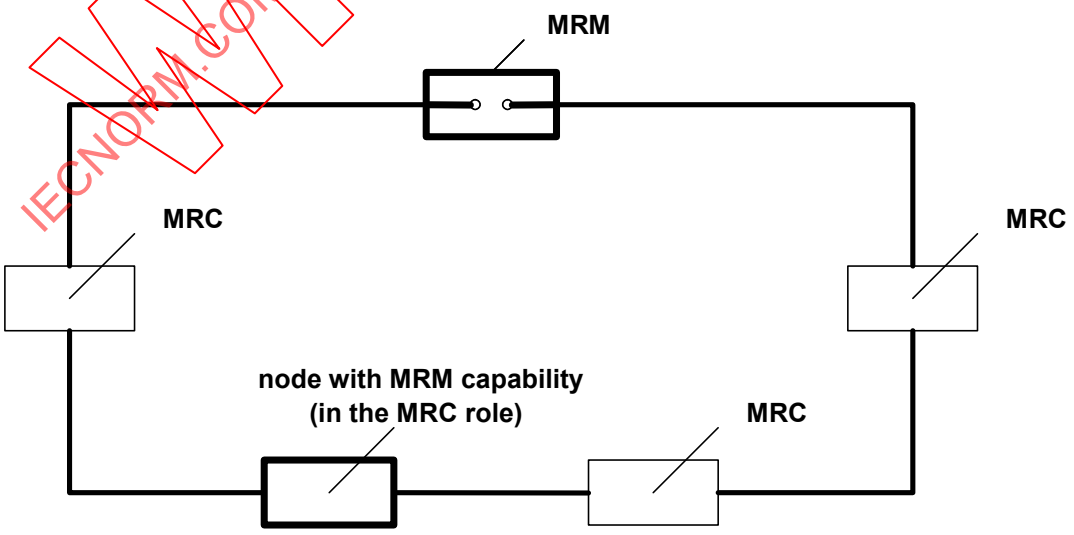


Figure 9 – More than one MRM in the ring

6.3.3.4.6 RT media redundancy (option)

This option supports bumpless media redundancy for RT Class 1 and RT Class 2 frames. The method is described in 6.3.8.2.3 for the attribute Redundancy of the IEEE 802.1D class. The Redundancy attribute of a port allows to transmit RT Class 1 and RT Class 2 frames via two associated ports in a ring topology. Thus, two identical frames are sent (same cycle information).

The MRM shall control the RT Ring State for RT media redundancy:

- When the media redundancy Ring State enters CLOSED state, the MRM shall periodically send RT test frames on ring ports in a determined time interval in both directions.
- As long as the MRM does not receives its own RT test frames (one ore more MRC in the ring have not entered RT Redundancy Mode), the MRM shall signal RT Ring State open condition and shall not activate the Redundancy attribute at MRM ring ports.
- While the MRM receives its own RT test frames, it shall activate the Redundancy attribute at its ring ports and signal RT Ring State closed condition (RT redundancy available, all MRC in the ring have entered RT Redundancy Mode).

Each MRC shall not forward RT test frames as long as RT Redundancy Mode is disabled. When RT Redundancy Mode is enabled, each MRC shall forward RT test frames between its ring ports.

6.3.3.5 Usage with diagnosis and alarms

Media redundancy events shall cause diagnosis events and alarm notifications if an application relationship is established and the attribute Check Media Redundancy has the value TRUE.

6.3.3.5.1 Media redundancy diagnosis dependencies

Figure 10 shows the dependencies of media redundancy diagnosis:

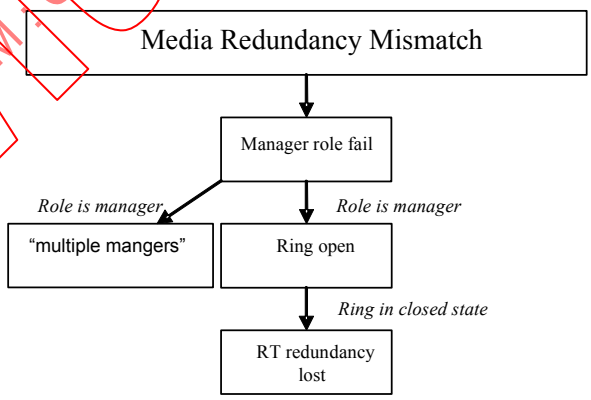


Figure 10 – Media redundancy diagnosis dependencies

- If a node in a redundancy domain is not in manager role, a "Manager role fail" diagnosis event shall be signalled. All other media redundancy diagnosis events are locked.

- If a node in a redundancy domain is in manager role and another manager is detected, the “multiple managers” event shall be signalled. This event may occur in parallel to ring state events “ring open” or “RT redundancy lost”.

- If a node in a redundancy domain is in manager role and an open ring is detected, the “ring open” event shall be signalled.

- If a node in a redundancy domain is in manager role, the option media redundancy for RT is activated and no open ring is detected, the “RT redundancy lost” event shall be signalled if RT redundancy is not available.

These events are indicated by means of the State Change service.

In addition, media redundancy handles the following events for both ring ports of a redundancy domain by means of the Neighborhood Change service:

- If the attribute Check Ringport Neighborhood is TRUE and the attribute Ring Port Domain State is BAD or UNDEFINED, the “Peer Domain ID mismatch” event shall be signalled.

- If the attributes Check Ringport Neighborhood and RT Redundancy Mode are both TRUE and the Ring Port RT State is BAD or UNDEFINED, the “Peer RT mismatch” event shall be signalled.

6.3.3.6 MRP parameters

6.3.3.6.1 Ring port parameters

Ring port parameterization for the MRM and all MRC in a ring shall comply with the settings from Table 60.

Table 60 – MRP network/connection parameters

Parameter	Value
Link speed	The link speed shall be at least 100 Mbit/s
Duplex setting	Ring ports shall operate in full duplex mode, Administrative mode of a port may be set to autonegotiation, but negotiated value (oper mode) shall be full duplex

6.3.3.6.2 Ring Topology parameters

The number of nodes participating in ring shall not exceed 50 nodes.

NOTE For more than this number of nodes in a ring, the maximum recovery time may be exceeded and the ring may become instable.

6.3.3.6.3 MRM and MRC parameters

The MRM defines with its parameter set the recovery time of a ring. Table 61 specifies the consistent set of parameter for a ring recovery time of 200 ms.

Table 61 – MRM parameters

Parameter	max. recovery time 200 ms	Meaning
MRP_TOPchgT	10 ms	Topology Change (Clear Address Table) request interval
MRP_TOPNRmax	3	Topology Change (Clear Address Table) repeat count
MRP_TSTshortT	10 ms	MRP_Test short interval
MRP_TSTdefaultT	20 ms	MRP_Test default interval
MRP_TSTNRmax	3	MRP_Test monitoring count

Table 62 specifies the MRC parameter set.

Table 62 – MRC parameters

Parameter	max. recovery time 200 ms	Meaning
MRP_LNKdownT	20 ms	Link Down Timer interval
MRP_LNKupT	20 ms	Link Up Timer interval
MRP_LNKNRmax	4	Link Change (Up or Down) count

NOTE Traffic load in the ring should not exceed 90%.

6.3.4 Real-time cyclic ASE

6.3.4.1 Overview

The real-time cyclic (RTC) is a concept of reading and writing cyclic data by means of buffered services. It uses IEEE 802.3 or UDP services.

6.3.4.2 RTC class specification

6.3.4.2.1 General

The RTC ASE defines for each CR one RTC object type.

6.3.4.2.2 Template

A RTC object is described by the following template:

ASE: **RTC ASE**
CLASS: **RTC**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:
1 (m) Key Attribute: CREP
SERVICES:
1 (m) OpsService: Set Prov Data
2 (m) OpsService: Set Prov Status
3 (m) OpsService: PPM Activate
4 (m) OpsService: Close
5 (m) OpsService: Start
6 (m) OpsService: Error
7 (m) OpsService: Get Cons Data
8 (m) OpsService: Get Cons Status
9 (m) OpsService: Set RedRole
10 (m) OpsService: CPM Activate
11 (m) OpsService: No Data

12 (m) OpsService: Stop

6.3.4.2.3 Attributes

CREP
This key attribute identifies the instance of the protocol machine to which the data belong.

Attribute type: Unsigned32

6.3.4.3 RTC service specification

6.3.4.3.1 Set Prov Data

This local service may be used to update the transmission buffer. Table 63 shows the parameters of the service.

Table 63 – Set Prov Data

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Data	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

Data
This parameter shall be used for data.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned8

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.4.3.2 Set Prov Status

This local service may be used to update the APDU status of the transmission buffer. Table 64 shows the parameters of the service.

Table 64 – Set Prov Status

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Status	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

Status
This parameter shall be used for the field APDU status.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned8

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.4.3.3 PPM Activate

This local service may be used to activate the provider protocol machine (PPM). Table 65 shows the parameters of the service.

Table 65 – PPM Activate

Parameter name	Req	Cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
VLAN Prio	M	
VLAN ID	M	
Soft Sync	M	
Reduction Ratio	M	
Phase	M	
Sequence	M	
Default Values	M	
Default Status	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

DA
This parameter shall be used for the destination MAC address.

SA
This parameter shall be used for the source MAC address.

Frame ID
This parameter shall be used for the Frame ID.

VLAN Prio
This parameter shall be set according to the requested parameter from the connect service.

VLAN ID
This parameter shall be set according to the requested parameter from the connect service.

Soft Sync
This parameter shall be set true if soft sync shall be used.

Reduction Ratio
This parameter shall be used for the Reduction Ratio.

Phase
This parameter shall be used for the Phase.

Sequence
This parameter shall be used for the Sequence.

Default Values
This parameter shall be used to determine the defaults.

Default Status
This parameter shall be used to determine the defaults.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned8

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.4.3.4 Close

This local service may be used to close the provider protocol machine (PPM). Table 66 shows the parameters of the service.

Table 66 – Close

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Result(+)		M
CREP		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

Result(+)
This parameter indicates that the service request succeeded.

6.3.4.3.5 Start

This local service may be used to indicate the first transmission of the provider protocol machine (PPM). Table 67 shows the parameters of the service.

Table 67 – Start

Parameter name	Ind
Argument	M
CREP	M

Argument
The argument shall convey the service specific parameters of the service indication.

CREP
This parameter is the local identifier for the desired CR.

6.3.4.3.6 Error

This local service may be used to indicate an error during transmission of the provider protocol machine (PPM). Table 68 shows the parameters of the service.

Table 68 – Error

Parameter name	Ind
Argument	M
CREP	M
No Data Send	M

Argument
The argument shall convey the service specific parameters of the service indication.

CREP
This parameter is the local identifier for the desired CR.

No Data Send
This parameter indicates the failure of data transmission.

6.3.4.3.7 Get Cons Data

This local service may be used to read data from the reception buffer. Table 69 shows the parameters of the service.

Table 69 – Get Cons Data

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M
Data		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

Result(+)
This parameter indicates that the service request succeeded.

Data
This parameter shall be used for data.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned8

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.4.3.8 Get cons status

This local service may be used to read the APDU status from the reception buffer. Table 70 shows the parameters of the service.

Table 70 – Get cons status

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M
Status		M
Recv Counter		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

Result(+)
This parameter indicates that the service request succeeded.

Status
This parameter shall be used for the APDU status.

Recv Counter
This parameter shall be used for the receive counter

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned8

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.4.3.9 Set RedRole

This local service may be used to set the redundancy status of the CR. Table 71 shows the parameters of the service.

Table 71 – Set RedRole

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Red Role	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

Red Role
This parameter determines the redundancy role (primary, secondary).

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned8

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.4.3.10 CPM activate

This local service may be used to activate the consumer protocol machine (CPM). Table 72 shows the parameters of the service.

Table 72 – CPM activate

Parameter name	Req	Cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
Prio	M	
VLAN	M	
Expected Length	M	
Start Ton	M	
Timeout Base	M	
Watchdog Factor	M	
Datahold Factor	M	
Default Value	M	
Default Status	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

DA
This parameter shall be used for the destination MAC address.

SA
This parameter shall be used for the source MAC address.

Frame ID
This parameter contains the value of the corresponding attribute of the CR ASE object.

Prio
This parameter contains the value of the corresponding attribute of the CR ASE object.

VLAN
This parameter contains the value of the corresponding attribute of the CR ASE object.

Expected Length
This parameter contains the value of the corresponding attribute of the CR ASE object.

Start Ton
This parameter shall be set to true if the timeout shall be monitored immediately. Otherwise it starts with the reception of the first frame.

Timeout Base
This parameter contains the value of the corresponding attribute of the CR ASE object.

Watchdog Factor
This parameter contains the value of the corresponding attribute of the CR ASE object.

Datahold Factor
This parameter contains the value of the corresponding attribute of the CR ASE object.

Default Value
This parameter contains the value of the corresponding attribute of the CR ASE object.

Default Status
This parameter contains the value of the corresponding attribute of the CR ASE object.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned8

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5 Real-time acyclic ASE

6.3.5.1 Overview

The real-time acyclic (RTA) is a concept of reading and writing acyclic data by means of acknowledged services. It uses IEEE 802.3 or UDP services.

6.3.5.2 RTA class specification

6.3.5.2.1 General

The RTA ASE defines for each CR one RTA object type:

6.3.5.2.2 Template

A RTA object is described by the following template:

ASE: **RTA ASE**
CLASS: **RTA**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	CREP
2	(m)	Attribute:	Protocol machine Parameter
2.1	(m)	Attribute:	DA
2.2	(m)	Attribute:	SA
2.3	(m)	Attribute:	Frame ID
2.4	(m)	Attribute:	VLAN Prio
2.5	(m)	Attribute:	VLAN ID
2.6	(m)	Attribute:	RTA Timeout Factor
2.7	(m)	Attribute:	M Retry
2.8	(m)	Attribute:	Send Seq Num
2.9	(m)	Attribute:	Ack Seq Num
2.10	(m)	Attribute:	Dst IP
2.11	(m)	Attribute:	Src IP
2.12	(m)	Attribute:	Transport

SERVICES:

1	(m)	OpsService:	APMS Activate
2	(m)	OpsService:	APMR Activate
3	(m)	OpsService:	AMPS A Data
4	(m)	OpsService:	APMR A Data
5	(m)	OpsService:	AMPR Ack
6	(m)	OpsService:	APMS Error
7	(m)	OpsService:	APMR Error
8	(m)	OpsService:	APMS Close
9	(m)	OpsService:	APMR Close

6.3.5.2.3 Attributes

CREP
This key attribute identifies the instance of the protocol machine to which the data belong.

Attribute type: Unsigned32

Protocol machine Parameter
This attribute contains the following attributes to describe the behavior of the protocol machines:

DA
This attribute contains the physical address of the destination device according to IEEE 802

Attribute type: OctetString[6]

SA
This attribute contains the physical address of the device according to IEEE 802.3

Attribute type: OctetString[6]

Frame ID
This attribute contains the identifier of the data within the frame.

Attribute type: Unsigned16

Allowed Values: ALARM_HIGH, ALARM_LOW

VLAN Prio
This attribute contains the priority of the frame according to IEEE 802.1Q.

Attribute type: Unsigned8

Allowed Values: LOW, HIGH

VLAN ID
This attribute contains the VLAN ID according to IEEE 802.1Q.

Attribute type: Unsigned16

Allowed Value: 0

RTA Timeout Factor
This attribute contains the timeout factor within a sent RTA Data frame has to be Acknowledged (ACK).

Attribute type: Unsigned16

Time Base: 100 ms

Allowed Values: 1 up to 100 mandatory, 101 to 2^{16}-1 optional

M Retry
This attribute contains the number of retrys if no ACK received within RTA Timeout Factor. The value range is from 3 to 15 retries.

Attribute type: Unsigned16

Send Seq Num
This attribute contains the Sender Sequence Number.

Attribute type: Unsigned16

Allowed Values: 0...0x7FFF, 0xFFFF, 0xFFFE

Ack Seq Num
This attribute contains the Acknoledge Sequence Number.

Attribute type: Unsigned16

Allowed Values: 0...0x7FFF, 0xFFFF, 0xFFFE

Dst IP
This attribute contains the destination IP address if UDP Services shall be used.

Attribute type: OctetString[4]

Src IP
This attribute contains the local IP address if UDP Services shall be used.

Attribute type: OctetString[4]

Transport
This attribute contains the requested conveyance.

Attribute type: Unsigned8

Allowed Values: RTC, UDP

6.3.5.3 RTA service specification

6.3.5.3.1 APMS Activate

This local service shall be used to activate the sender protocol machine (APMS). Table 73 shows the parameters of the service.

Table 73 – APMS Activate

Parameter name	Req	Cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
VLAN Prio	M	
VLAN ID	M	
RTA Timeout Factor	M	
M Retry	M	
Transport	M	
Dst IP	O	
Src IP	O	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request. These parameters shall correspond with the ASE attributes.

CREP
This parameter identifies the instance of the protocol machine.

DA
This parameter shall be used for the physical address of the destination device according to IEEE 802.3.

SA
This parameter shall be used for the physical address of the device according to IEEE 802.3.

Frame ID
This parameter shall be used for the Frame ID.

VLAN Prio
This parameter shall contain the priority of the frame according to IEEE 802.1Q

VLAN ID
This parameter shall contain the VLAN ID according to IEEE 802.1Q.

RTA Timeout Factor
This parameter shall contain the timeout factor.

M Retry
This parameter shall contain the numer of retrys.

Dst IP
This parameter shall contain the destination IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Src IP
This parameter shall contain the local IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Transport
This parameter shall contain the requested conveyance.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned16

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.2 APMR Activate

This local service shall be used to activate the receiver protocol machine (APMR). Table 74 shows the parameters of the service.

Table 74 – APMR Activate

Parameter name	Req	Cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
VLAN Prio	M	
VLAN ID	M	
Transport	M	
Dst IP	O	
Src IP	O	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument
The argument shall convey the service specific parameters of the service request. These parameters shall correspond with the ASE attributes.

CREP
This parameter identifies the instance of the protocol machine.

DA
This parameter shall be used for the physical address of the destination device according to IEEE 802.3.

SA
This parameter shall be used for the physical address of the device according to IEEE 802.3.

Frame ID
This parameter shall be used for the Frame ID.

VLAN Prio
This parameter shall contain the priority of the frame according to IEEE 802.1Q

VLAN ID
This parameter shall contain the VLAN ID according to IEEE 802.1Q.

Transport
This parameter shall contain the requested conveyance.

Dst IP
This parameter shall contain the destination IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Src IP
This parameter shall contain the local IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Class
This parameter selects the error class.

Type: Unsigned16

Allowed Value: CTXT

Error code
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.3 APMS A Data

This service shall be used to send RTA Data. After a RTA Data frame was sent, it has to be acknowledged within a spezified time (based on RTA Timeout Factor). If no acknowledge (ACK) arrives the data frame will be resend a spezified number of times (M Retry). If no ACK arrived at all a APMS Error will be indicated (TIMEOUT). The communication uses sequence numbers to ensure proper sequence of data. Both communication endpoints have to be setup for communication to ensure proper sequence number setup. Only one service can be handled at a time. So the service has to be finished before a new one can be issued (INVALID_STATE result shall be returned if not finished). Table 75 shows the parameters of the service.

Table 75 – APMS A Data

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Data	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
ERRCLS		M
ERRCODE		M

Argument
The argument shall convey the service specific parameters of the service request. These parameters shall correspond with the ASE attributes.

CREP
This parameter identifies the instance of the protocol machine.

Data
This parameter shall be used for RTA-SDU data to be send. The length is limited by ADPU size.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

ERRCLS
This parameter selects the error class.

Type: Unsigned16

Allowed Value: CTXT

ERRCODE
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.4 APMR A Data

This local service may be used to receive RTA data. Table 76 shows the parameters of the service.

Table 76 – APMR A Data

Parameter name	Ind
Argument	M
CREP	M
Data	M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter identifies the instance of the protocol machine.

Data
This parameter contains the received RTA-SDU data

Result(+)
This parameter indicates that the service request succeeded.

6.3.5.3.5 APMR Ack

This service shall be used to send a RTA ACK after reception of a RTA Data. After RTA Data was received it has to be acknowledged by sending an ACK. Sending the ACK signals the sender that the Data was successfully received and further data may be send. If the ACK will not be send, the sender retrys and will timeout if no ACK arrives at all. Only one service call can be handled at a time. So the service has to be finished before a new one can be issued (INVALID_STATE result is set in not finished). Table 77 shows the parameters of the service.

Table 77 – APMR Ack

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
ERRCLS		M
ERRCODE		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter identifies the instance of the protocol machine.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

ERRCLS
This parameter selects the error class.

Type: Unsigned16

Allowed Value: CTXT

ERRCODE
This parameter selects the error code.

Type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.6 APMS error

This local service may be used to indicate an APMS error. Table 78 shows the parameters of the service.

Table 78 – APMS Error

Parameter name	Ind
Argument	M
CREP	M
ERRCLS	M
ERRCODE	M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter identifies the instance of the protocol machine.

ERRCLS
This parameter selects the error class.

Type: Unsigned16

Allowed Value: see Table 79

ERRCODE
This parameter selects the error code.

Type: Unsigned16

Allowed Value: see Table 79

Table 79 – APMS Error ERRCLS/ERRCODE

ERRCLS	ERRCODE	Meaning
PROTOCOL	TIMEOUT	Transmission timeout occurred
PROTOCOL	SEQNUM	Sequence number mismatch
PROTOCOL	LMPM	LMPM error

6.3.5.3.7 APMR Error

This local service may be used to indicate an APMR error. Table 80 shows the parameters of the service.

Table 80 – APMR Error

Parameter name	Ind
Argument	M
CREP	M
ERRCLS	M
ERRCODE	M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter identifies the instance of the protocol machine.

ERRCLS
This parameter selects the error class.

Type: Unsigned16

Allowed Value: see Table 81

ERRCODE
This parameter selects the error code.

Type: Unsigned16

Allowed Value: see Table 81

Table 81 – APMR Error ERRCLS/ERRCODE

ERRCLS	ERRCODE	Meaning
PROTOCOL	SEQNUM	Sequence number of received data mismatch
PROTOCOL	LMPM	LMPM error

6.3.5.3.8 APMS_Close

This local service shall be used to close the Sender protocol machine (APMS). Table 82 shows the parameters of the service.

Table 82 – APMS_Close

Parameter name	Req	Cnf
Argument	M	
CREP	M	
ErrCode	U	
Result(+)		S
CREP		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter identifies the instance of the protocol machine.

ErrCode
This optional parameter selects the error code.

Type: Unsigned16

Allowed Value: See corresponding ASE attribute

Result(+)
This parameter indicates that the service request succeeded.

6.3.5.3.9 APMR_Close

This local service shall be used to close the Receiver protocol machine (APMS). Table 83 shows the parameters of the service.

Table 83 – APMR_Close

Parameter name	Req	Cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter identifies the instance of the protocol machine.

Result(+)
This parameter indicates that the service request succeeded.

6.3.6 Remote procedure call ASE

6.3.6.1 Overview

The remote procedure call (RPC) concepts according to Publication C706 of the The Open Group shall be applied. This ASE defines the special usage of the Publication C706 of the The Open Group as a transport protocol.

6.3.6.2 RPC class specification

6.3.6.2.1 Template

The RPC object is described by the following template:

ASE: **RPC ASE**
CLASS: **RPC**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:
1 (m) Key Attribute: Implicit
SERVICES:
1 (m) OpsService: Connect
2 (m) OpsService: Release
3 (m) OpsService: Read
4 (m) OpsService: Write
5 (m) OpsService: Control

6.3.6.2.2 Attributes

Implicit
The attribute Implicit indicates that the object is implicitly addressed by the service.

6.3.6.3 RPC service specification

6.3.6.3.1 Connect

This confirmed service shall be used to establish a connection by means of CL RPC. Table 84 shows the parameters of the service.

Table 84 – Connect

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Args Max
This parameter shall contain the maximum buffer size available for the response.

Type: Unsigned32

Args Len
This parameter shall contain the actual number of octets for the parameter Args.

Type: Unsigned32

Args
This parameter shall contain the RPC user data.

Type: Octet String

Result(+)
This parameter indicates that the service request succeeded.

PNIO Status
This parameter shall be set to zero.

Type: Unsigned32

Result(-)
This parameter indicates that the service request failed.

PNIO Status
This parameter shall be set according to the current error and shall consist of the elements Error code = "IODConnectRes", Error Decode = "PNIO", Error code 1, and Error code 2. Error code 1 and Error code 2 shall contain the actual error code according to the protocol definition.

Type: Unsigned32

6.3.6.3.2 Release

This confirmed service shall be used to release a connection by means of CL RPC. Table 85 shows the parameters of the service.

Table 85 – Release

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Args Max
This parameter shall contain the maximum buffer size available for the response.

Type: Unsigned32

Args Len
This parameter shall contain the actual number of octets for the parameter Args.

Type: Unsigned32

Args
This parameter shall contain the RPC user data.

Type: Octet String

Result(+)
This parameter indicates that the service request succeeded.

PNIO Status
This parameter shall be set to zero.

Type: Unsigned32

Result(-)
This parameter indicates that the service request failed.

PNIO Status
This parameter shall be set according to the current error and shall consist of the elements Error code = “IODReleaseRes”, Error Decode = “PNIO”, Error code 1, and Error code 2. Error code 1 and Error code 2 shall contain the actual error code according to the protocol definition.

Type: Unsigned32

6.3.6.3.3 Read

This confirmed service shall be used to convey a higher layer read service by means of CL RPC. Table 86 shows the parameters of the service.

Table 86 – Read

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Args Max
This parameter shall contain the maximum buffer size available for the response.

Type: Unsigned32

Args Len
This parameter shall contain the actual number of octets for the parameter Args.

Type: Unsigned32

Args
This parameter shall contain the RPC user data.

Type: Octet String

Result(+)
This parameter indicates that the service request succeeded.

PNIO Status
This parameter shall be set to zero.

Type: Unsigned32

Result(-)
This parameter indicates that the service request failed.

PNIO Status
This parameter shall be set according to the current error and shall consist of the elements Error code = "IODReadRes", Error Decode = "PNIORW", Error code 1, and Error code 2. Error code 1 and Error code 2 shall contain the actual error code according to the protocol definition.

Type: Unsigned32

6.3.6.3.4 Write

This confirmed service shall be used to convey a higher layer write service by means of CL RPC. Table 87 shows the parameters of the service.

Table 87 – Write

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Args Max
This parameter shall contain the maximum buffer size available for the response.

Type: Unsigned32

Args Len
This parameter shall contain the actual number of octets for the parameter Args.

Type: Unsigned32

Args
This parameter shall contain the RPC user data.

Type: Octet String

Result(+)
This parameter indicates that the service request succeeded.

PNIO Status
This parameter shall be set to zero.

Type: Unsigned32

Result(-)
This parameter indicates that the service request failed.

PNIO Status
This parameter shall be set according to the current error and shall consist of the elements Error code = "IODWriteRes", Error Decode = "PNIORW", Error code 1, and Error code 2. Error code 1 and Error code 2 shall contain the actual error code according to the protocol definition.

Type: Unsigned32

6.3.6.3.5 Control

This confirmed service shall be used to convey a higher layer "End of Parameter" and "Application Ready" service by means of CL RPC. Table 88 shows the parameters of the service.

Table 88 – Control

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Args Max
This parameter shall contain the maximum buffer size available for the response.

Type: Unsigned32

Args Len
This parameter shall contain the actual number of octets for the parameter Args.

Type: Unsigned32

Args
This parameter shall contain the RPC user data.

Type: Octet String

Result(+)
This parameter indicates that the service request succeeded.

PNIO Status
This parameter shall be set to zero.

Type: Unsigned32

Result(-)
This parameter indicates that the service request failed.

PNIO Status
This parameter shall be set according to the current error and shall consist of the elements Error code = "IOXControlRes", Error Decode = "PNIO", Error code 1, and Error code 2. Error code 1 and Error code 2 shall contain the actual error code according to the protocol definition.

Type: Unsigned32

6.3.7 Link layer discovery ASE

6.3.7.1 Overview

The concepts according to IEEE 802.1AB standard shall be applied. This part of the specification defines the special usage of Chassis ID, Port ID, Chassis ID Subtype, Port ID Subtype, and Time to live. Furthermore, extensions for the OUI regarding frames and MIB are defined.

Furthermore, the RFC 3490 (IDNA) shall be applied for the content syntax of Chassis ID and Port ID.

If SNMP is supported, the system shall conform to the LLDP management specifications in and shall implement the sections of the basic LLDP MIB for the operating mode being

implemented. If SNMP is not supported, the system shall provide storage and retrieval capability equivalent to the functionality for the operating mode being implemented.

6.3.7.2 IEEE 802.1AB class specification

6.3.7.2.1 Template

The IEEE 802.1AB object is described by the following template:

ASE: **Link Layer Discovery ASE**
CLASS: **IEEE 802.1AB**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	Chassis ID
3	(m)	Attribute:	Port ID
3	(m)	Attribute:	LLDP Time To Live
4	(o)	Attribute:	Port Description
5	(o)	Attribute:	System Name
6	(o)	Attribute:	System Description
7	(o)	Attribute:	System Capabilities
8	(m)	Attribute:	Management Address
9	(o)	Attribute:	Object Identifier
10	(o)	Attribute:	List of Organizationally-specific extensions
10.1	(m)	Attribute:	Organizationally Unique Identifier
10.2	(c)	Constraint:	Organizationally Unique Identifier = 00-80-C2
10.2.1	m)	Attribute:	List of ISO/IEC 8802.1 Subtypes
10.2.1.1	(m)	Attribute:	Subtype
10.2.1.2	(c)	Constraint:	Subtype = PORT_VLAN_ID
10.2.1.2.1	(m)	Attribute:	Port VLAN Identifier
10.2.1.3	(c)	Constraint:	Subtype = PORT_AND_PROTOCOL_VLAN_ID
10.2.1.3.1	(m)	Attribute:	Flags
10.2.1.3.2	(m)	Attribute:	PPVID Reference Number
10.2.1.4	(c)	Constraint:	Subtype = VLAN_NAME
10.2.1.4.1	(m)	Attribute:	VLAN ID
10.2.1.4.2	(m)	Attribute:	VLAN Name
10.2.1.5	(c)	Constraint:	Subtype = PROTOCOL_IDENTITY
10.2.1.5.1	(m)	Attribute:	List of Protocol Identities
10.2.1.5.1.1	(m)	Attribute:	Protocol Identity
10.3	(c)	Constraint:	Organizationally Unique Identifier = 00-12-0F
10.3.1	(m)	Attribute:	List of ISO/IEC 8802.3 Subtypes
10.3.1.1	(m)	Attribute:	Subtype
10.3.1.2	(c)	Constraint:	Subtype = MAC_PHY_CONFIGURATION_STATUS
10.3.1.2.1	(m)	Attribute:	Auto Negotiation Support And Status
10.3.1.2.2	(m)	Attribute:	Advertised Capability
10.3.1.2.3	(m)	Attribute:	Operational MAU type
10.3.1.3	(c)	Constraint:	Subtype = POWER_VIA_MDI
10.3.1.3.1	(m)	Attribute:	MDI Power Support
10.3.1.3.2	(m)	Attribute:	Power Pair
10.3.1.3.3	(m)	Attribute:	Power Class
10.3.1.4	(c)	Constraint:	Subtype = LINK_AGGREGATION
10.3.1.4.1	(m)	Attribute:	Link Aggregation Status
10.3.1.4.2	(m)	Attribute:	Aggregated Port ID

10.3.1.5	(c)	Constraint:	Subtype = MAXIMUM_FRAME_SIZE
10.3.1.5.1	(m)	Attribute:	Maximum Frame Size
10. 4	(c)	Constraint:	Organizationally Unique Identifier = 00-0E-CF
10. 4.1	(m)	Attribute:	List of PNIO Subtypes
10. 4.1.1	(m)	Attribute:	Subtype
10. 4.1.2	(c)	Constraint:	Subtype = LLDP_PNIO_DELAY
10. 4.1.2.1	(m)	Attribute:	PnioDelay
10. 4.1.2.1.1	(m)	Attribute:	PortRxDelayLocal
10. 4.1.2.1.2	(m)	Attribute:	PortRxDelayRemote
10. 4.1.2.1.3	(m)	Attribute:	PortTxDelayLocal
10. 4.1.2.1.4	(m)	Attribute:	PortTxDelayRemote
10. 4.1.2.1.5	(m)	Attribute:	CableDelayLocal
10. 4.1.3	(c)	Constraint:	Subtype = LLDP_PNIO_PORTSTATUS
10. 4.1.3.1	(m)	Attribute:	PnioPortStatus
10. 4.1.3.1.1	(m)	Attribute:	RTClass2PortStatus
10. 4.1.3.1.2	(m)	Attribute:	RTClass3PortStatus
10.14.4.1.3.1.2.1	(m)	Attribute:	Status
10.14.4.1.3.1.2.2	(m)	Attribute:	Mode
10. 4.1.4	(c)	Constraint:	Subtype = LLDP_PNIO_ALIAS
10. 4.1.4.1	(m)	Attribute:	PnioAlias
10.14.4.1.5	(c)	Constraint:	Subtype = LLDP_PNIO_MRP_PORT_STATUS
10.14.4.1.5.1	(m)	Attribute:	PnioMRPPortStatus
10.14.4.1.5.1.1	(m)	Attribute:	MRPDomainUUID
10.14.4.1.5.1.2	(m)	Attribute:	MRRTPortStatus
10.14.4.1.6	(c)	Constraint:	Subtype = LLDP_PNIO_INTERFACE_MAC_ADDRESS
10.14.4.1.6.1	(m)	Attribute:	PnioInterfaceMacAddress
10.14.4.1.7	(c)	Constraint:	Subtype = LLDP_PNIO_PTCP_STATUS
10.14.4.1.7.1	(m)	Attribute:	PnioPTCPStatus
10.14.4.1.7.1.1	(m)	Attribute:	PnioPTCPMasterSourceAddress
10.14.4.1.7.1.2	(m)	Attribute:	PnioPTCPSubdomainUUID
10.14.4.1.7.1.3	(m)	Attribute:	PnioIRDataUUID
10.14.4.1.7.1.4	(m)	Attribute:	PnioLengthOfPeriod
10.14.4.1.7.1.4.1	(m)	Attribute:	Length
10.14.4.1.7.1.4.2	(m)	Attribute:	Validity
10.14.4.1.7.1.5	(m)	Attribute:	PnioRedPeriodBegin
10.14.4.1.7.1.6	(m)	Attribute:	PnioOrangePeriodBegin
10.14.4.1.7.1.7	(m)	Attribute:	PnioGreenPeriodBegin
11	(m)	Attribute:	List of Remote Systems Data
11.1	(m)	Attribute:	Port ID
11.2	(m)	Attribute:	Chassis ID
11.3	(m)	Attribute:	MAC Address
11.4	(m)	Attribute:	LLDP Time To Live
11.5	(o)	Attribute:	Port Description
11.6	(o)	Attribute:	System Name
11.7	(o)	Attribute:	System Description
11.8	(o)	Attribute:	System Capabilities
11.9	(o)	Attribute:	Management Address
11.10	(o)	Attribute:	Object Identifier
11.11	(m)	Attribute:	Propagation Delay Factor
11.12	(m)	Attribute:	List of Organizationally-specific extensions
11.12.1	(m)	Attribute:	Organizationally Unique Identifier
11.12.2	(c)	Constraint:	Organizationally Unique Identifier = 00-80-C2
11.12.2.1	(m)	Attribute:	List of ISO/IEC 8802.1 Subtypes

11.12.2.1.1	(m)	Attribute:	Subtype
11.12.2.1.2	(c)	Constraint:	Subtype = PORT_VLAN_ID
11.12.2.1.2.1	(m)	Attribute:	Port VLAN Identifier
11.12.2.1.3	(c)	Constraint:	Subtype = PORT_AND_PROTOCOL_VLAN_ID
11.12.2.1.3.1	(m)	Attribute:	Flags
11.12.2.1.3.2	(m)	Attribute:	PPVID Reference Number
11.12.2.1.4	(c)	Constraint:	Subtype = VLAN_NAME
11.12.2.1.4.1	(m)	Attribute:	VLAN ID
11.12.2.1.4.2	(m)	Attribute:	VLAN Name
11.12.2.1.5	(c)	Constraint:	Subtype = PROTOCOL_IDENTITY
11.12.2.1.5.1	(m)	Attribute:	List of Protocol Identities
11.12.2.1.5.1.1	(m)	Attribute:	Protocol Identity
11.12.3	(c)	Constraint:	Organizationally Unique Identifier = 00-12-0F
11.12.3.1	(m)	Attribute:	List of ISO/IEC 8802.3 Subtypes
11.12.3.1.1	(m)	Attribute:	Subtype
11.12.3.1.2	(c)	Constraint:	Subtype = MAC_PHY_CONFIGURATION_STATUS
11.12.3.1.2.1	(m)	Attribute:	Auto Negotiation Support And Status
11.12.3.1.2.2	(m)	Attribute:	Advertised Capability
11.12.3.1.2.3	(m)	Attribute:	Operational MAU type
11.12.3.1.3	(c)	Constraint:	Subtype = POWER_VIA_MDI
11.12.3.1.3.1	(m)	Attribute:	MDI Power Support
11.12.3.1.3.2	(m)	Attribute:	Power Pair
11.12.3.1.3.3	(m)	Attribute:	Power Class
11.12.3.1.4	(c)	Constraint:	Subtype = LINK_AGGREGATION
11.12.3.1.4.1	(m)	Attribute:	Link Aggregation Status
11.12.3.1.4.2	(m)	Attribute:	Aggregated Port ID
11.12.3.1.5	(c)	Constraint:	Subtype = MAXIMUM_FRAME_SIZE
11.12.3.1.5.1	(m)	Attribute:	Maximum Frame Size
11.12.4	(c)	Constraint:	Organizationally Unique Identifier = 00-0E-CF
11.12.4.1	(m)	Attribute:	List of PNIO Subtypes
11.12.4.1.1	(m)	Attribute:	Subtype
11.12.4.1.2	(c)	Constraint:	Subtype = LLDP_PNIO_DELAY
11.12.4.1.2.1	(m)	Attribute:	PnioDelay
11.12.4.1.2.1.1	(m)	Attribute:	PortRxDelayLocal
11.12.4.1.2.1.2	(m)	Attribute:	PortRxDelayRemote
11.12.4.1.2.1.3	(m)	Attribute:	PortTxDelayLocal
11.12.4.1.2.1.4	(m)	Attribute:	PortTxDelayRemote
11.12.4.1.2.1.5	(m)	Attribute:	CableDelayLocal
11.12.4.1.3	(c)	Constraint:	Subtype = LLDP_PNIO_PORTSTATUS
11.12.4.1.3.1	(m)	Attribute:	PnioPortStatus
11.12.4.1.3.1.1	(m)	Attribute:	RTClass2PortStatus
11.12.4.1.3.1.2	(m)	Attribute:	RTClass3PortStatus
11.12.4.1.3.1.2.1	(m)	Attribute:	Status
11.12.4.1.3.1.2.2	(m)	Attribute:	Mode
11.12.4.1.4	(c)	Constraint:	Subtype = LLDP_PNIO_ALIAS
11.12.4.1.4.1	(m)	Attribute:	PnioAlias
11.12.4.1.5	(c)	Constraint:	Subtype = LLDP_PNIO_MRP_PORT_STATUS
11.12.4.1.5.1	(m)	Attribute:	PnioMRPPortStatus
11.12.4.1.5.1.1	(m)	Attribute:	MRPDomainUUID
11.12.4.1.5.1.2	(m)	Attribute:	MRRTPortStatus
11.12.4.1.6	(c)	Constraint:	Subtype = LLDP_PNIO_INTERFACE_MAC_ADDRESS
11.12.4.1.6.1	(m)	Attribute:	PnioInterfaceMacAddress
11.12.4.1.7	(c)	Constraint:	Subtype = LLDP_PNIO_PTCP_STATUS

11.12.4.1.7.1	(m)	Attribute:	PnioPTCPStatus
11.12.4.1.7.1.1	(m)	Attribute:	PnioPTCPMasterSourceAddress
11.12.4.1.7.1.2	(m)	Attribute:	PnioPTCPSubdomainUUID
11.12.4.1.7.1.3	(m)	Attribute:	PnioIRDataUUID
11.12.4.1.7.1.4	(m)	Attribute:	PnioLengthOfPeriod
11.12.4.1.7.1.4.1	(m)	Attribute:	Length
11.12.4.1.7.1.4.2	(m)	Attribute:	Validity
11.12.4.1.7.1.5	(m)	Attribute:	PnioRedPeriodBegin
11.12.4.1.7.1.6	(m)	Attribute:	PnioOrangePeriodBegin
11.12.4.1.7.1.7	(m)	Attribute:	PnioGreenPeriodBegin

SERVICES:

1	(m)	OpsService:	Remote Systems Data Change

6.3.7.2.2 Attributes

Implicit
The attribute Implicit indicates that the object is implicitly addressed by the service.

Chassis ID
This non-volatile attribute contains the name of the interface which shall be used as Chassis ID subtype locally assigned compliant with IEEE 802.1AB. This field shall contain an octet string indicating the specific identifier for the particular chassis in this system. Because Chassis ID and Port ID are concatenated to form the local MSAP identifier, the value chosen for Chassis ID shall be non-null. The value of the attribute shall be equal to the value Real Station Name and Interface Name.

NOTE 1 This attribute can be written via DCP services (Name of Station).

Attribute type: VisibleString[240]

Port ID
This attribute contains the name of the local port which shall be used as Port ID subtype locally assigned compliant with IEEE 802.1AB. The value shall be “port-xyz” or “port-xyz-rstuv” where x,y,z is in the range “0”-“9” from 001 up to 255 and r, s, t, u, v is in the range “0”-“9” from 00 000 up to 65 535. The values x,y,z shall be used to identify the port number. The values r,s,t,u,v shall be used to identify the slot. The values “port-001” or “port-001-00000” shall be used for the first port submodule for an interface within slot 0.

Attribute type: OctetString[8] or OctetString[14]

NOTE 2 Within the context of LLDP protocol a DTE is referred to as port. It does not mean a port from the UDP context.

LLDP Time To Live
This attribute indicates the number of seconds that the recipient LLDP agent is to regard the information associated with this chassis and port identifier to be valid. It shall be compliant with IEEE 802.1AB.

Attribute type: Unsigned16

Allowed Values: 20

Port Description
This attribute contains an alpha-numeric string that indicates the port description. It shall be compliant with IEEE 802.1AB.

NOT 3E If IETF RFC 2863 is implemented, the use of the ifDescr object is recommended for this field.

System Name
This attribute contains an alpha-numeric string that indicates the system’s administratively-assigned name. It shall be compliant with IEEE 802.1AB.

NOTE 4 This should be the system’s fully-qualified domain name. If implementations support IETF RFC 3418, the use of the sysName object is recommended for this field.

System Description
This attribute contains an alpha-numeric string that is the textual description of the network entity. It shall be compliant with IEEE 802.1AB.

NOTE 5 This value should include the full name and version identification of the system's hardware type, software operating system, and networking software. If implementations support IETF RFC 3418, the use of the sysDescr object is recommended for this field.

System Capabilities

This attribute contains a bit-map of the capabilities that define the primary function(s) of the system. The bit positions for each function and the associated MIB or standard that are likely, but not guaranteed, to be supported are listed in Table 89. A binary one in the associated bit indicates the existence of that capability. It shall be compliant with IEEE 802.1AB.

NOTE 6 Individual systems may indicate more than one implemented functional capability (for example, both a bridge and router capability).

Table 89 – System capabilities

Position	Meaning
Bit 0	Other
Bit 1	Repeater
Bit 2	Bridge
Bit 3	WLAN Access Point
Bit 4	Router
Bit 5	Telephone
Bit 6	DOC SIS cable device
Bit 7	Station Only

Management Address

This attribute contains an octet string indicating the particular management address. It shall be compliant with IEEE 802.1AB.

NOTE 7 The returned address should be the most appropriate for management use, typically a layer-3 address such as the IPv4 address, 192.168.254.10.

Object Identifier

This attribute contains an OID that identifies the type of hardware component or protocol entity associated with the indicated management address. If no OID is available, this field shall not be provided. It shall be compliant with IEEE 802.1AB.

NOTE 8 The interface number and OID are included to assist NMS discovery by indicating Enterprise-specific or other starting points for the search, such as the Interface (see RFC 2863) or Entity MIB (see RFC 2737).

List of Organizationally-specific extensions

This attribute list contains the following attributes:

Organizationally Unique Identifier

This attribute contains an OUI that is provided to allow different organizations, such as ISO/IEC 8802.1, ISO/IEC 8802.3, IETF, as well as individual software and equipment vendors, to define TLVs that advertise information to remote entities.

List of ISO/IEC 8802.1 Subtypes

This attribute list contains the following attributes:

Subtype

This attribute shall be complient with IEEE 802.1AB.

Allowed values: PORT_VLAN_ID, PORT_AND_PROTOCOL_VLAN-ID, VLAN_NAME, PROTOCOL_IDENTITY

Port VLAN Identifier

This attribute contains the VLAN ID for the bridge port as defined in IEEE 802.1Q. A value of zero shall be used if the system either does not know the PVID or does not support port-based VLAN operation.

Flags

This attribute contains a bit map indicating the port and protocol VLAN capability and status.

Allowed Values: PPVID_SUPPORTED, PPVID_ENABLED

PPVID Reference Number

This attribute contains the PPVID number for this 802 LAN station.

NOTE 9 If the port is not capable of supporting port and protocol VLANs and/or the port is not enabled with any port protocol VLAN, the PPVID number should be zero.

VLAN ID
This attribute contains the VID number associated with the VLAN name.

VLAN Name
This attribute contains the VLAN's name.

Attribute type: OctetString[32]

NOTE 10 If implementations support IETF RFC 2674, the use of the dot1QVLANStaticName object is recommended for this field.

List of Protocol Identies
This attribute list contains the following attributes:

Protocol Identity
This attribute contains the first n octets of the protocol after the layer-2 addresses (i.e., starting with the length/type field) that the sender would like to advertise. The value of n is determined by the need for the protocol to disambiguate itself. The protocol information string shall include enough octets to allow the receiver to correctly identify the protocol and its version.

NOTE 11 Spanning tree protocols, for example, would need to include five octets: LLC address (two octets), Protocol ID (two octets), plus the protocol version (one octet).

List of ISO/IEC 8802.3 Subtypes
This attribute list contains the following attributes:

Subtype
This attribute shall be complient with IEEE 802.1AB.

Allowed values: MAC_PHY_CONFIGURATION_STATUS, POWER_VIA_MDI, LINK_AGGREGATION, MAXIMUM_FRAME_SIZE

Auto Negotiation Support And Status
This attribute contains a bit map that identifies the auto-negotiation support and current status of the local ISO/IEC 8802.3 LAN station as defined in Table 90. If auto-negotiation support bit (bit-0) is one and the auto-negotiation status bit (bit-1) is zero, the ISO/IEC 8802.3 physical media dependent sublayer operating mode will be determined the default configuration field value rather than by auto-negotiation.

Table 90 – Auto negotiation support and status

Position	Meaning
Bit 0	Autonegotiation Support (0 – not supported, 1 – supported)
Bit 1	Autonegotiation Status (0 – disabled, 1 – enabled)

NOTE 12 This attribute can be written via Write Adjusted Port Data.

Advertised Capability
This attribute contains an integer value as defined by the ifMauAutoNegCapAdvertisedBits object in IETF RFC 3636.

Operational MAU type
This attribute is an integer value indicating the operational MAU type of the sending device. This value is derived from the list position of the corresponding dot3MauType as listed in IETF RFC 3636 (or subsequent revisions) and is equal to the last number in the respective dot3MauType OID. For MAU types not listed in RFC 3636 (or subsequent revisions), the value of this field shall be set to zero.

MDI Power Support
This attribute contains a bit-map of the MDI power capabilities and status as defined in Table 91.

Table 91 – MDI Power Support

Position	Meaning
Bit 0	Port Class (0 – Power Sourcing Equipment, 1 – Powered Device)
Bit 1	MDI Power Support (0 – not supported, 1 – supported)

Position	Meaning
Bit 2	MDI Power State (0 – disabled, 1 – enabled)
Bit 3	Pairs Control Ability (0 - pair selection can not be controlled, 1 - pair selection can be controlled)

Power Pair
This attribute contains an integer value as defined by the pethPsePortPowerPairs object in IETF RFC 3621.

Power Class
This attribute contains an integer value as defined by the pethPsePortPowerClassification object in IETF RFC 3621.

Link Aggregation Status
This attribute contains a bit map of the link aggregation capabilities and the current aggregation status of the link as defined in Table 92.

Table 92 – Link aggregation status

Position	Meaning
Bit 0	Aggregation Capability (0 – not supported, 1 – supported)
Bit 1	Aggregation Status (0 – disabled, 1 – enabled)

Aggregated Port ID
This attribute contains the ISO/IEC 15802 aggregated port identifier. An AggPortID is derived from the ifNumber in the ifIndex for the interface.

Maximum Frame Size
This attribute contains an integer value indicating the maximum supported frame size in octets shall be set to 1 522.

List of PNIO Subtypes
This attribute list contains the following attributes:

Subtype
This attribute contains the following values.

Allowed values: LLDP_PNIO_DELAY, LLDP_PNIO_PORTSTATUS, LLDP_PNIO_ALIAS, LLDP_PNIO_MRP_PORT_STATUS, LLDP_PNIO_INTERFACE_MAC_ADDRESS, LLDP_PNIO_PTCP_STATUS

PnioDelay
This attribute contain of the following attributes:

PortRxDelayLocal
This attribute contains the local receiving delay of the port in nanoseconds.

Attribute type: Unsigned32

Allowed values: 1-0xFFF

PortRxDelayRemote
This attribute contains the local receiving delay of the peer port in nanoseconds.

Attribute type: Unsigned32

Allowed values: 1-0xFFF

PortTxDelayLocal
This attribute contains the local transmission delay of the port in nanoseconds.

Attribute type: Unsigned32

Allowed values: 1-0xFFF

PortTxDelayRemote
This attribute contains the local transmission delay of the peer port in nanoseconds.

Attribute type: Unsigned32

Allowed values: 1-0xFFF

CableDelayLocal
This attribute contains the local cable delay in nanoseconds.

Attribute type: Unsigned32

Allowed values: 1-0xFFFFF

PnioPortStatus
This attribute contain of the following attributes:

RTClass2PortStatus
This attribute contains the port status for RT_CLASS_2.

Attribute type: Unsigned16

Allowed values: NOT_USED, CONFIGURED, ORANGE_ACTIVE

RTClass3PortStatus
This attribute contains the following attributes.

Status
This attribute contains the port status for RT_CLASS_3.

Allowed values: NOT_USED, IRDATA_CONFIGURED, RTACLASS3_NEXT_ACTIVE, RT

Mode
This attribute contains the port mode for RT_CLASS_3.

Allowed values: STANDARD, OPTIMISED

PnioAlias
This attribute contains the following octet string:

"Value of attribute Port ID""."Value of Chassis ID"

Attribute type: OctetString

PnioMRPPortStatus
This attribute shall consist of the following attributes.

MRPDomainUUID
This attribute contains the MRP domain UUID.

Attribute type: UUID

MRRTPortStatus
This attribute contains the port status for MRRT.

Allowed values: MRRT_NOT_USED, MRRT_CONFIGURED, MRRT_ACTIVE

PnioInterfaceMacAddress
This attribute contains the IEEE 802 MAC address.

Attribute type: OctetString[6]

PnioPTCPStatus
This attribute shall consist of the following attributes.

PnioPTCPMasterSourceAddress
This attribute contains the IEEE 802 MAC address of the PTCP master.

Attribute type: OctetString[6]

PnioPTCPSubdomainUUID
This attribute contains the UUID of the PTCP subdomain.

Attribute type: UUID

PnioIRDataUUID
This attribute contains the UUID of the IR data.

Attribute type: UUID

PnioLengthOfPeriod
This attribute shall consist of the following attributes.

Length
This attribute contains the value for the duration of the cycle in nanoseconds. The value shall be a multiply of 31 250 nanoseconds.

Attribute type: Unsigned32

Allowed values: 0x00007A12 - 0x003D0900 if valid, 0 if invalid

Validity
This attribute contains the validity of the length.

Allowed values: INVALID, VALID

PnioRedPeriodBegin
This attribute shall consist of the following attributes.

Offset
This attribute contains the value for the offset related to the begin of the cycle in nanoseconds.

Attribute type: Unsigned32

Allowed values: 0x00000000 - 0x003D08FF if valid, 0 if invalid

Validity
This attribute contains the validity of the offset.

Allowed values: INVALID, VALID

PnioOrangePeriodBegin
This attribute shall consist of the following attributes.

Offset
This attribute contains the value for the offset related to the begin of the cycle in nanoseconds.

Attribute type: Unsigned32

Allowed values: 0x00000000 - 0x003D08FF if valid, 0 if invalid

Validity
This attribute contains the validity of the offset.

Allowed values: INVALID, VALID

PnioGreenPeriodBegin
This attribute shall consist of the following attributes.

Offset
This attribute contains the value for the offset related to the begin of the cycle in nanoseconds.

Attribute type: Unsigned32

Allowed values: 0x00000000 - 0x003D08FF if valid, 0 if invalid

Validity
This attribute contains the validity of the offset.

Allowed values: INVALID, VALID

List of Remote Systems Data
This attribute list contains the same attributes as specified for Real List of Ports including all subsequent attributes until Maximum Frame Size. Furthermore, the following additional attributes are included.

Port ID
This attribute has the same definition as Own Port ID.

Chassis ID
This attribute has the same definition as Interface Name.

MAC Address
This attribute contains the devices physical address according to IEEE 802.3 MAC address.

Attribute type: OctetString[6]

Propagation Delay Factor
This attribute contains the measured line delay. The time base shall be 1 ns.

Attribute type: Unsigned32

6.3.7.2.3 Invocation of the IEEE 802.1AB object

For the invocation of the IEEE 802.1AB object the following rules apply:

– IEEE 802.1AB objects shall exist for each interface.

6.3.7.3 IEEE 802.1AB service specification

The protocol according to IEEE 802.1AB shall be used to send and receive the attribute values of the ASE. IEEE 802.1AB uses the services of the LMPM to transmit and receive information to and from other LLDP agents.

6.3.7.3.1 Remote systems data change

This local service shall be used to indicate a change of certain Remote Systems Data. Table 93 shows the parameters of the service.

Table 93 – Remote systems data change

Parameter name	Ind
Argument	M
Local Port ID	M
List of Neighbours	U
Remote Chassis ID	U
Remote Port ID	U
Remote Cable Delay	U
Remote RT_CLASS_3 Port Status	U
Remote RT_CLASS_2 Port Status	U
Remote PTCP Status	U
Remote Line Delay	U
Auto Negotiation Support And Status	U
Advertised Capability	U
Operational MAU type	U

Argument
The argument shall convey the service specific parameters of the service indication.

Local Port ID
This parameter is the local identifier.

List of Neighbours
This parameter list consist of the following parameter:

Remote Chassis ID
This parameter contains the value of the attribute of the ASE.

Remote Port ID
This parameter contains the value of the attribute of the ASE.

Remote Cable Delay
This parameter contains the value of the attribute of the ASE.

Remote RT_CLASS_3 Port Status
This parameter contains the value of the attribute of the ASE.

Remote RT_CLASS_2 Port Status
This parameter contains the value of the attribute of the ASE.

Remote PTCP Status
This parameter contains the value of the attribute of the ASE.

Remote Line Delay
This parameter contains the value of the sum of the following attribute of the ASE:

CableDelayLocal

PortRxDelayLocal

PortTxDelayRemote

Auto Negotiation Support And Status
This parameter contains the value of the attribute of the ASE.

Advertised Capability
This parameter contains the value of the attribute of the ASE.

Operational MAU type
This parameter contains the value of the attribute of the ASE.

6.3.7.3.2 Placeholder

This is an empty subclause to be in line with IEC directive part 2 and the structure of service specifications.

6.3.8 MAC bridges ASE

6.3.8.1 Overview

The concepts according to IEEE 802.1D shall be applied. This clause defines the utilization of the IEEE 802.1D.

The bridging mechanisms of IEEE 802.1D shall be temporary disabled within the RT_CLASS_3 phase. In this case, the bridging of RT_CLASS_3 frames shall be done according to the attributes defined by this ASE with the behavior according to the RT_CLASS_3 Relay protocol machine. The values of the attributes define a special forwarding table.

Furthermore, special forwarding rules are defined for media redundancy (see 6.3.2.4 and attribute redundancy in 6.3.8.2.3) for RT_CLASS_2 and RT_CLASS_1.

6.3.8.2 IEEE 802.1D class specification

6.3.8.2.1 General

The IEEE 802.1D ASE defines one IEEE 802.1D object type:

6.3.8.2.2 Template

An IEEE 802.1D object is described by the following template:

ASE:	**IEEE 802.1D ASE**		
CLASS:	**IEEE 802.1D**		
CLASS ID:	**not used**		
PARENT CLASS:	**top**		
ATTRIBUTES:			
1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	IR Global Data
2.1	(m)	Attribute:	IR Data ID
2.2	(o)	Attribute:	Max Bridge Delay
2.3	(o)	Attribute:	Number of Ports
2.4	(o)	Attribute:	List of Port Delays
2.4.1	(m)	Attribute:	Max Port Tx Delay

2.4.2	(m)	Attribute:	Max Port Rx Delay
3	(m)	Attribute:	List of IR Frame Data Elements
3.1	(m)	Attribute:	Frame Send Offset
3.2	(m)	Attribute:	Data Length
3.3	(m)	Attribute:	Reduction Ratio
3.4	(m)	Attribute:	Phase
3.5	(m)	Attribute:	Frame ID
3.6	(m)	Attribute:	Ethertype
3.7	(m)	Attribute:	Rx Port
3.8	(m)	Attribute:	Frame Details
3.8.1	(m)	Attribute:	Sync Frame
3.8.2	(m)	Attribute:	Meaning Frame Send Offset
3.9	(m)	Attribute:	Tx Port Group
3.9.1	(m)	Attribute:	Number Of Tx Port Group Array Elements
3.9.2	(m)	Attribute:	Tx Port Group Array
4	(m)	Attribute:	List of Ports
4.1	(m)	Attribute:	Port
4.1.1	(m)	Attribute:	Transmission Period Sending
4.1.2	(m)	Attribute:	Transmission Period Receiving
4.1.3	(m)	Attribute:	Redundancy
4.1.4	(m)	Attribute:	Local Port State
4.1.5	(m)	Attribute:	Local RTClass 3 Port State
4.1.6	(m)	Attribute:	Local RTClass 2 Port State
5	(o)	Attribute:	Clock Domain Phase Counter
SERVICES:			
1	(m)	OpsService:	Port State Change
2	(o)	OpsService:	Set Port State
3	(o)	OpsService:	Flush filtering data base
4	(m)	OpsService:	IFW IRT Schedule Add
5	(m)	OpsService:	IFW IRT Schedule Remove
6	(m)	OpsService:	IFW Schedule

6.3.8.2.3 Attributes

Implicit
The attribute Implicit indicates that the IEEE 802.1D object is implicitly addressed by the service.

IR Global Data
This attribute contains the following attributes:

IR Data ID
This attribute contains a unique identifier to identify the generation of the IR Data unambiguously.

Attribute type: UUID

Max Bridge Delay
This optional attribute contains the value for the maximum bridge delay in nanoseconds.

Attribute type: Unsigned32

Allowed Values: 0 - unknown, 0x00000001-0x3B9AC9FF - values calcutated by engineering tool

Number of Ports
This optional attribute contains the value for the number of ports containing the attributes Max Port Tx Delay and Max Port Rx Delay.

Attribute type: Unsigned32

Allowed Values: 0x00000001-0x000000FF

List of Port Delays
This optional attribute contains the Port Delay Elements. A list element includes the following attributes:

Max Port Tx Delay
This optional attribute contains the value for the maximum transmission delay for the port in nanoseconds.

Attribute type: Unsigned32

Allowed Values: 0 - unknown, 0x00000001-0x3B9AC9FF - values calcutated by engineering tool

Max Port Rx Delay
This optional attribute contains the value for the maximum delay for receiver of the port in nanoseconds.

Attribute type: Unsigned32

Allowed Values: 0 - unknown, 0x00000001-0x3B9AC9FF - values calcutated by engineering tool

List of IR Frame Data Elements
This attribute contains the IR Frame Data Elements. A list element includes the following attributes:

Frame Send Offset
This attribute contains the send or receive offset in relation to start of the cycle. The time base is according to the RT_CLASS_3 Time Base.

Attribute type: Unsigned32

Data Length
This attribute contains the length of data including C_SDU length plus APDU_Status length.

Attribute type: Unsigned16

Allowed Values for RT_CLASS_3: 4 to 1 440

Reduction Ratio
This attribute contains the reduction ratio of the send clock.

Attribute type: Unsigned16

The allowed values are shown in Table 94:

Table 94 – Allowed values of ReductionRatio

Value(decimal)	Meaning
1	Mandatory
2	Mandatory
4	Mandatory
8	Mandatory
16	Mandatory
1 024 - 65 535	Reserved
other	Optional

Phase
This attribute contains the particular cycle for the frame.

Attribute type: Unsigned16

Allowed Values: 1 until current value of the attribute Reduction Ratio

Frame ID
This attribute contains the identifier of the frame which is configured by project planning.

Attribute type: Unsigned32

The allowed values according to Table 95 shall be used.

Table 95 – Frame IDs for RT_CLASS_3

Value (hexadecimal)	Meaning
0x0080	Dedicated to synchronization
0x0100 – 0x7FFF	Dedicated to RT_CLASS_3 unicast and multicast

Ethertype
This attribute contains the value of the type/Length fields of the DLPDU.

Attribute type: Unsigned16

Allowed Values: 0x8892

Rx Port
This attribute contains the number of the receiving port.

Attribute type: Unsigned8

Allowed Values: LOCAL_INJECTION, PORT_1 to PORT_255

Frame Details
This attribute consists of the following elements:

Attribute type: Unsigned8

Sync Frame
This field shall identify a sync frame.

Allowed values shall be according to Table 96.

Table 96 – Sync Frame

Meaning
NO_SYNC_FRAME
PRIMARY_SYNC_FRAME
SECONDARY_SYNC_FRAME

FrameSendOffset
This attribute contains the interpretation of the attribute FrameSendOffset with the values according to Table 97.

Table 97 – FrameSendOffset

Value	Meaning
TRANSMISSON_TIME	Field FrameSendOffset specifies the point of time for receiving or transmitting a frame
RT_CLASS_3_BEGIN	Field FrameSendOffset specifies the beginning of the RT_CLASS_3 interval within a phase
RT_CLASS_3_END	Field FrameSendOffset specifies the ending of the RT_CLASS_3 interval within a phase

Tx Port Group
This attribute consists of the following elements:

Number Of Tx Port Group Array Elements
This field shall count the successing Tx Port Group Array entries.

Allowed values are 1,3,5,7,9,11,13,15, ..., 31,33.

Attribute type: Unsigned8

Tx Port Group Array
The Tx Port Group Array is an array of octets that shall contain at least one and at most 33 octets referred to as Tx Port Group Array entry. A Tx Port Group Array Element shall consist of at least one and at most 8 TxPort entries referred to as TxPortGroup entry 0 to TxPortGroup entry 7. Therefore, the number of TxPortGroup octet corresponds to the number of ports within a device and shall be calculated as follows

1) $N = M_{highest}$ DIV 8 +1

where

N is the number of TxPortGroup octets or the number of array elements,

$M_{highest}$ is the highest number of TxPorts within a device (maximum 255).

The unused TxPort entries shall be set to zero.

NOTE 1 The term DIV stands for division without rest. The term MOD stands for the rest of the division.

Bit 0: TxPortEntry_0
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 0 is present. A padding bit shall be used if no TxPort is present.

Table 98 – Tx Port Entry

Value (hexadecimal)	Meaning
0x00	Transmission off
0x01	Transmission on

The TxPort of local injection is always placed in TxPortEntry_1 of the TxPortGroup octet number one.

Bit 1: TxPortEntry_1
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 1 is present. A padding bit shall be used if no TxPort is present.

Bit 2: TxPortEntry_2
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 2 is present. A padding bit shall be used if no TxPort is present.

Bit 3: TxPortEntry_3
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 3 is present. A padding bit shall be used if no TxPort is present.

Bit 4: TxPortEntry_4
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 4 is present. A padding bit shall be used if no TxPort is present.

Bit 5: TxPortEntry_5
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 5 is present. A padding bit shall be used if no TxPort is present.

Bit 6: TxPortEntry_6
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 6 is present. A padding bit shall be used if no TxPort is present.

Bit 7: TxPortEntry_7
This bit shall be set with the values according to Table 98 if the TxPort with the number that meets m MOD 8 = 7 is present. A padding bit shall be used if no TxPort is present.

Where m is the number of the current TxPort with $1 \leq m \leq N$.

List of Ports
This attribute shall consist of the following elements:

Port
This attribute consists of the following elements:

Attribute type: Unsigned8

Redundancy
This attribute contains the value of the current redundancy setting. The value NoRedundancy means that the IEEE 802.1D rules shall be applied.

The value Port1 to Port255 means that RT_CLASS_1 and RT_CLASS_2 frames shall be transmitted via two associated ports in a ring topology. A node may act as a generating node, forwarding node, or receiving node, depicted in Figure 11.

The generating node shall transparent duplicate RT_CLASS_1 and RT_CLASS_2 frames and shall convey those frames also via blocked ports.

The forwarding node shall forward received RT_CLASS_1 and RT_CLASS_2 frames via the associated port, even if this port is blocked. However, the received addresses shall not be integrated in the forwarding address table (FDB, filtering data base) to avoid influencing the transmission of all other frames.

The receiving node shall received RT_CLASS_1 and RT_CLASS_2 frames even via blocked ports. However, the received addresses shall not be integrated in the forwarding address table to avoid influencing the transmission of all other frames.

All nodes shall discard RT_CLASS_1 and RT_CLASS_2 frames if the IEEE 802.3 source MAC address is the own one.

NOTE 2 In future versions real time redundancy will be also applied to RTA frames

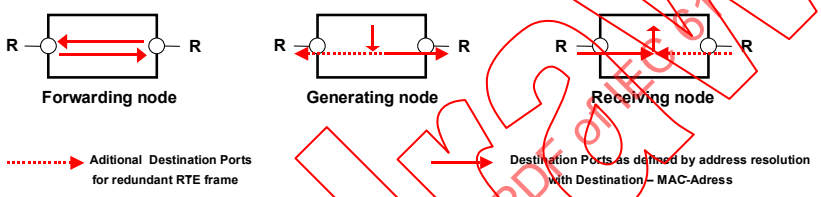


Figure 11 – Locating the destination for redundant RT frames

NOTE 3 The receiving CPM is not aware of this mechanism. It just maintains the cycle counter and monitoring time. Received duplicate frames do not influence the operation.

Attribute type: Unsigned8

Allowed Values: NoRedundancy, Port1 – Port255

Transmission Period Sending

This attribute contains the value of the current transmission period of the port in sending direction. The scheduling of the sending process according IEEE 802.1Q ASE shall be applied.

The following transmitting rules for RT_CLASS_3 shall be applied:

The local queuing process shall transmit a frame according to Frame Send Offset and Tx Port Group Array if the calculation of Phase and Reduction Ratio fits Phase Number.

The following forwarding rules for RT_CLASS_3 shall be applied:

The local queuing process shall transmit a prior received frame according to Frame Send Offset, Data Length, Frame ID, and Rx Port. If the calculation of Phase and Reduction Ratio fits Phase Number, then the frame shall be transmitted on all ports stored in Tx Port Group Array.

Within the YELLOW period the forwarding rules of IEEE 802.1D shall be omitted.

The following transmitting rules shall be applied:

The local queuing process shall transmit a frame according if it's size is shorter then the remaining YELLOW interval.

The following forwarding rules shall be applied:

The local queuing process shall transmit a prior received frame if it's size is shorter then the remaining YELLOW interval.

Within the ORANGE, and GREEN period, the IEEE 802.1D shall be applied.

The transmission period shall be controlled by a local clock. This clock shall be synchronized over the network using RED or ORANGE periods and may be synchronized over the network using only GREEN periods. The duration of all periods shall not exceed the time value calculated by means of the attribute Send Clock Factor, which shall be defined by other ASE. The accuracy is determined by conformance classes.

Attribute type: Unsigned8

Allowed Values: RED (optional), ORANGE (optional), GREEN, YELLOW (optional)

EXAMPLE As shown in Figure 12, the sending of frames is divided into different periods at the local port. Each phase is defined by $T_{Sendclock}$ which is between 31,25 µs and 4 ms. The number of frames of each period depends on the scheduled frames for the current phase and may be also zero.

Transmission Period Receiving
This attribute contains the value of the current transmission period of the port in receving direction.

Within the RED period, only in time RT_CLASS_3 DLPDUs and in time PTCP Sync DLPDUs shall be forwarded according RED Relay rules. All other received DLPDUs shall be handeled by the MAC Relay according to the forwarding rules of IEEE 802.1D.

The following receiving rules for RT_CLASS_3 shall be applied:

The local queuing process shall receive a frame according to Frame Send Offset, Data Length, Frame ID, Rx Port and the calculation of Phase and Reduction Ratio fits Phase Number.

Phase := (Clock Domain Phase Counter MOD Reduction ratio) + 1 (19).

The received IEEE 802.3 source MAC addresses shall not be integrated in the forwarding address table to avoid influencing the transmission of all other frames.

The following receiving rules shall be applied:

Within the ORANGE, and GREEN period, the IEEE 802.1D (MAC Relay) shall be applied.

The transmission period shall be controlled by a local clock. This clock shall be synchronized over the network using RED or ORANGE periods and may be synchronized over the network using only GREEN periods. The duration of all periods shall not exceed the time value calculated by means of the attribute Send Clock Factor, which shall be defined by other ASE. The accuracy is determined by conformance classes.

Attribute type: Unsigned8

Allowed Values: RED (optional), ORANGE (optional), GREEN

Local Port State
This attribute contains the value for the port according IEEE 802.1D bridge management.

Attribute type: Unsigned8

Allowed Values: DISCARDING, BLOCKED, FORWARDING

Local RTClass 3 Port State
This attribute contains the value for the port according to the RED phase. The value OFF means not used or configured for RED phase or the preconditions could not be reached so far. The value RTCLASS3_UP means that the transmission direction is in the RED phase. The value RTCLASS3_RUN means that both directions are in the RED phase.

Attribute type: Unsigned8

Allowed Values: OFF, RTCLASS3_UP, RTCLASS3_RUN

Local RTClass 2 Port State
This attribute contains the value for the port according to the ORANGE phase. The value OFF means not used or configured for ORANGE phase or the preconditions could not be reached so far. The value RTCLASS2_RUN means that transmitter and receiver are in the ORANGE phase.

Attribute type: Unsigned8

Allowed Values: OFF, RTCLASS2_RUN

Clock Domain Phase Counter
This attribute contains the count of passed phases. All nodes used with RT_CLASS_3 shall use the same value. It is gained by PTCP clock synchronization.

Attribute type: Unsigned64

6.3.8.3 IEEE 802.1D service specification

6.3.8.3.1 Port state change

This local service shall be used to indicate a change of certain Local Systems Data. Table 99 shows the parameters of the service.

Table 99 – Port state change

Parameter name	Ind
Argument	M
Local Port ID	M
Local Port State	M
Local RTClass 3 Port State	M
Local RTClass 2 Port State	M

Argument
The argument shall convey the service specific parameters of the service indication.

Local Port ID
This parameter is the value of the corresponding attribute of the ASE object.

Local Port State
This parameter is the value of the corresponding attribute of the ASE object.

Local RTClass 3 Port State
This parameter is the value of the corresponding attribute of the ASE object.

Local RTClass 2 Port State
This parameter is the value of the corresponding attribute of the ASE object.

6.3.8.3.2 Set port state

This local service shall be used to set a certain Local Systems Data. Table 100 shows the parameters of the service.

Table 100 – Set port state

Parameter name	Req
Argument	M
Local Port ID	M
Local Port State	U
Local RTClass 3 Port State	U
Local RTClass 2 Port State	U

Argument
The argument shall convey the service specific parameters of the service indication.

Local Port ID
This parameter is the value of the corresponding attribute of the ASE object.

Local Port State
This parameter is the value of the corresponding attribute of the ASE object.

Local RTClass 3 Port State
This parameter is the value of the corresponding attribute of the ASE object.

Local RTClass 2 Port State
This parameter is the value of the corresponding attribute of the ASE object.

6.3.8.3.3 Flush filtering data base

This local service shall be used to flush the local filtering data base. Table 101 shows the parameters of the service.

Table 101 – Flush filtering data base

Parameter name	Req
Argument	M

Argument
The argument shall convey the service specific parameters of the service indication.

6.3.8.3.4 IFW IRT Schedule Add

The Schedule Add service is used to store ASE attributes by means of a local service.

Table 102 shows the parameter of the service.

Table 102 – IFW IRT Schedule Add

Parameter name	Req	Cnf
Argument	M	
CREP	M	
D_Port	M	
Reduction Ratio	M	
Phase	M	
Result(+)		M
CREP		M
D_Port		M
Status		M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

D_Port
This parameter contains the Port ID.

Reduction Ratio
This parameter contains the value for the equivalent attribute of the CRL class specification.

Phase
This parameter contains the value for the equivalent attribute of the CRL class specification.

Result(+)
This parameter indicates that the service request succeeded.

Status
This parameter shall contain the value OK.

6.3.8.3.5 IFW IRT Schedule Remove

The Schedule Remove service is used to remove the schedule by means of a local service.

Table 103 shows the parameter of the service.

Table 103 – IFW IRT Schedule Remove

Parameter name	Req	Cnf
Argument	M	
D_Port	M	
Result(+)		M
CREP		M
D_Port		M
Status		M

Argument
The argument shall convey the service specific parameters of the service request.

D_Port
This parameter contains the Port ID.

Result(+)
This parameter indicates that the service request succeeded.

CREP
This parameter is the local identifier for the desired CR.

Status
This parameter shall contain the value OK.

6.3.8.3.6 IFW Schedule

The IFW Schedule service is used to set the phase by means of a local service.

Table 104 shows the parameter of the service.

Table 104 – IFW Schedule

Parameter name	Req	Cnf
Argument	M	
Phase	M	
Len	M	
Result(+)		M
Status		M

Argument
The argument shall convey the service specific parameters of the service request.

Phase
This parameter contains the phase.

Len
This parameter contains the duration of the phase.

Result(+)
This parameter indicates that the service request succeeded.

Status
This parameter shall contain the value OK.

6.3.8.4 Invocation of the IEEE 802.1D object

For the invocation of the IEEE 802.1D object the following rules apply:

- IEEE 802.1D objects shall exist for each interface
- The services with the IFW prefix shall exist for each RT_CLASS_3 interface

6.3.9 Virtual bridged LAN ASE

6.3.9.1 Overview

The concepts according to IEEE 802.1D standard shall be applied. This part of the specification defines the utilization of the IEEE 802.1Q standard. It defines extensions for RT_CLASS_3 and RT_CLASS_2 regarding priorization of frames.

Application processes contain data that are transmitted in different priorities according IEEE 802.1Q. Furthermore, the usage of these priorities shall be extended in different time intervals according to this specification. The IEEE 802.1Q ASE defines attributes to determine the timing behavior.

The formal model of the IEEE 802.1Q ASE is presented by the IEEE 802.1Q class specification, containing a description of its attributes.

6.3.9.2 IEEE 802.1Q class specification

6.3.9.2.1 Template

An IEEE 802.1Q object is described by the following template:

ASE:	**IEEE 802.1Q ASE**		
CLASS:	**IEEE 802.1Q**		
CLASS ID:	**not used**		
PARENT CLASS:	**TOP**		
ATTRIBUTES:			
1	(m)	Key Attribute:	not used
2	(o)	Attribute:	Network Control High Queue
3	(m)	Attribute:	Network Control Low Queue
4	(m)	Attribute:	Prio 7 Send Queue
5	(m)	Attribute:	Prio 6
5.1	(o)	Attribute:	Red Send Queue
5.2	(o)	Attribute:	Orange Send Queue
5.3	(m)	Attribute:	Prio 6 Send Queue
6	(m)	Attribute:	Prio 5 Send Queue
7	(m)	Attribute:	Prio 4 Send Queue
8	(m)	Attribute:	Prio 3 Send Queue
9	(m)	Attribute:	Prio 2 Send Queue
10	(m)	Attribute:	Prio 1 Send Queue
11	(m)	Attribute:	Prio 0 Send Queue
12	(m)	Attribute:	Transmission Period
13	(o)	Attribute:	Clock Domain Phase Counter

6.3.9.2.2 Attributes

This key attribute is not used.

Network Control High Queue
This optional attribute contains the list of DLPDUs for synchronization for those DLPDUs which are not planned within IR Data records or for those received outside the planned time. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Network Control Low Queue
This attribute contains the list of DLPDUs for IEEE 802.1D organizational priority for management protocols like LLDP, PTCP line delay measurement, MRP, MRRT and others. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 7 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 7. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 6
This attribute consists of the following attributes:

Red Send Queue
This optional attribute contains the list of DLPDUs for RT_CLASS_3 and for DLPDUs for synchronization which were planned within a IR Data record and received in time. The sequence of frames and the exact sending time shall be determined by other ASE attributes. After transmission the DLPDU shall be removed from the queue. With the start of a new phase, the Red Send Queue shall be filled with the DLPDUs for the current phase.

Attribute type: Octet String

NOTE The application layer for decentralized periphery defines a list of IR Frame Data elements to determine the sequence and exact sending time of frames.

Orange Send Queue
This optional attribute contains the list of DLPDUs for RT_CLASS_2 and for those RT_CLASS_3 DLPDUs which were received not in time. After transmission the DLPDU shall be removed from the queue. With the start of a new phase, the RT_CLASS_2 Send Queue shall be filled with the DLPDUs for the current phase.

Attribute type: Octet String

Prio 6 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 6 inclding DLPDUs for RT_CLASS_1 and RT_CLASS_UDP. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 5 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 5. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 4 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 4. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 3 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 3. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 2 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 2. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 1 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 1. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Prio 0 Send Queue
This attribute contains the list of DLPDUs for IEEE 802.1Q priority 0. After transmission the DLPDU shall be removed from the queue.

Attribute type: Octet String

Transmission Period
This attribute contains the value of the current transmission period of the port. It is aimed to control the usage of the bandwidth within the network.

NOTE In theory, each device could use always wired speed for sending frames. Such behavior applied over a certain amount of time would bring the system out operation. Therefore, a fair mechanism to restrict the sending performance of each device is provided.

Within the RED period, only DLPDUs from the Red Send Queue shall be transmitted.

Within the ORANGE period, only DLPDUs from the RED period, Network Control High Queue, the Orange Send Queue, and the Network Control Low Queue shall be transmitted.

Within the GREEN period, DLPDUs from all queues shall be transmitted according to their priorities. Within priority 6 the sending order Orange Send Queue and last Prio 6 Send Queue shall be applied.

Within the GREEN period, for the local scheduling the following rules shall be applied:

- The local queuing process shall drop all RT_CLASS_2/1 frames for the next phase if at the beginning of a new GREEN period the Orange/Prio 6 Send Queue is not empty. Each drop shall be signaled to the local application.
- In average, not more than 60 percent of bandwidth shall be used to avoid overload of the network.

Within the YELLOW period, DLPDUs from all queues shall be transmitted according to their priorities if the transmission of the frame ends within this period. If the calculated transmission time for a frame exceeds the period, the first frame with lesser priority can be tested optionally.

The sequence of periods shall be RED, ORANGE, GREEN, and YELLOW. The RED and/or ORANGE period may be omitted. The YELLOW period shall exist if RED and/or ORANGE exists and shall be omitted otherwise.

The transmission period shall be controlled by a local clock. This clock shall be synchronized over the network using RED or ORANGE periods and may be synchronized over the network using only GREEN periods. The duration of all periods shall not exceed the time value calculated by means of the attribute Send Clock Factor, which shall be defined by other ASE. The accuracy is determined by conformance classes.

Attribute type: Unsigned8

Allowed Values: RED (optional), ORANGE (optional), GREEN, YELLOW (optional)

EXAMPLE As shown in Figure 12, the sending of frames is divided into different periods at the local port. Each phase is defined by $T_{Sendclock}$ which is between 31,25 µs and 4 ms. The number of frames of each period depends on the scheduled frames for the current phase and may be also zero.

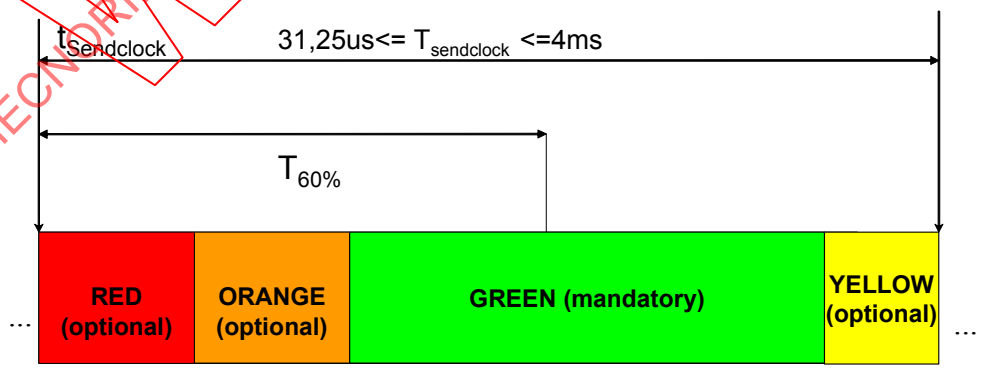


Figure 12 – Example of periods at a local port

The first and the last invocation of the RED period shall be synchronized between the local and the neighborhood port by means of Remote Systems Data Change (Remote RT_CLASS_3 Port Status).

Clock Domain Phase Counter
This attribute contains the count of passed phases. All nodes used with RT_CLASS_3 shall use the same value. It is gained by PTCP clock synchronization.

Attribute type: Unsigned64

6.3.9.2.3 Invocation of the IEEE 802.1Q object

For the invocation of the IEEE 802.1Q object the following rules apply:

- IEEE 802.1Q objects shall exist for each port.

6.3.9.3 IEEE 802.1Q service specification

There are no services specified.

6.3.10 Medium access ASE

6.3.10.1 Overview

The concepts according to IEEE 802.3 shall be applied. This clause defines the utilization of the IEEE 802.3.

6.3.10.2 IEEE 802.3 class specification

6.3.10.2.1 General

The IEEE 802.3 ASE defines one IEEE 802.3 object type.

6.3.10.2.2 Template

An IEEE 802.3 object is described by the following template:

ASE: **IEEE 802.3 ASE**
CLASS: **IEEE 802.3**
CLASS ID: **not used**
PARENT CLASS: **top**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	List of Ports
2.1	(m)	Attribute:	Port
2.1.1	(m)	Attribute:	MAU type
2.1.2	(m)	Attribute:	Autonegotiation Status
2.1.3	(m)	Attribute:	Status

SERVICES:

1	(m)	OpsService:	MAU type Change
2	(m)	OpsService:	Set MAU type

6.3.10.2.3 Attributes

Implicit
The attribute Implicit indicates that the IEEE803 object is implicitly addressed by the service.

List of Ports
This attribute shall consist of the following elements:

Port
This attribute consists of the following elements:

Attribute type: Unsigned8

MAU type
This attribute contains the value of the current MAU type according IETF RFC 3636. If the value of the attribute Status is DOWN or OFF, the MAU type shall be zero.

Attribute type: Unsigned16

Autonegotiation status
This attribute contains the value of the current autonegotiation setting. For optical transmission the value shall be set to DISABLE.

Attribute type: Unsigned8

Allowed Values: ENABLE, DISABLE

Status
This attribute contains the value of the current port status setting. It shall be OFF, if the media attachment unit is disabled. It shall be DOWN, if the media attachment unit is enabled but no Link is detected. It shall be UP, if the media attachment unit is enabled and a Link is detected.

Attribute type: Unsigned8

Allowed Values: UP, DOWN, OFF

6.3.10.3 IEEE 802.3 service specification

6.3.10.3.1 MAU type Change

This local service shall be used to indicate a change of certain Local Systems Data. Table 105 shows the parameters of the service.

Table 105 – MAU type change

Parameter name	Ind
Argument	M
Local Port ID	M
Local MAU type	U
Local Autonegotiation State	U
Local Link Status	U

Argument
The argument shall convey the service specific parameters of the service indication.

Local Port ID
This parameter is the value of the corresponding attribute of the ASE object.

Local MAU type
This parameter is the value of the corresponding attribute of the ASE object.

Local Autonegotiation Status
This parameter is the value of the corresponding attribute of the ASE object.

Local Link Status
This parameter is the value of the corresponding attribute of the ASE object.

6.3.10.3.2 Set MAU type

This local service shall be used to indicate a change of certain Remote Systems Data.Table 106 shows the parameters of the service.

Table 106 – Set MAU type

Parameter name	Ind
Argument	M
Local Port ID	M
Local MAU type	U
Local Autonegotiation State	U
Local Link Status	U

Argument
The argument shall convey the service specific parameters of the service indication.

Local Port ID
This parameter is the value of the corresponding attribute of the ASE object.

Local MAU type
This parameter is the value of the corresponding attribute of the ASE object.

Local Autonegotiation Status
This parameter is the value of the corresponding attribute of the ASE object.

Local Status
This parameter is the value of the corresponding attribute of the ASE object.

6.3.10.4 Invocation of the IEEE 802.3 object

For the invocation of the IEEE 802.3 object the following rules apply:

– IEEE 802.3 objects shall exist for each interface.

6.3.11 IP suite ASE

6.3.11.1 Overview

The utilization of the RFC 768 (UDP), RFC 791 (IP), RFC 792 (ICMP), RFC 826 (ARP), RFC 1112 (IP Multicasting) standards are defined for this specification. It includes definitions for UDP ports and IP multicast addresses and utilization of IP header fields for RT over UDP.

This ASE shall contain current volatile values of operation. Permanent values in non volatile memory shall be a matter of the DCP ASE defined in 6.3.1.

6.3.11.2 IP suite class specification

6.3.11.2.1 General

The IP suite ASE defines one Physical Device Management object type:

6.3.11.2.2 Template

A IP suite object is described by the following template:

ASE: **IP suite ASE**
CLASS: **IP suite**
CLASS ID: **not used**
PARENT CLASS: **top**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	IP
2.1	(m)	Attribute:	IP Address
2.2	(m)	Attribute:	Subnet Mask
2.3	(m)	Attribute:	Standard Gateway
3	(m)	Attribute:	ARP
3.1	(m)	Attribute:	Cache Size
3.2	(m)	Attribute:	Cache Timeout
3.3	(m)	Attribute:	List of Cache Entries
3.3.1	(m)	Attribute:	IEEE 802.3 MAC Address
3.3.2	(m)	Attribute:	IP Address
4	(m)	Attribute:	List of IP Multicast addresses
4.1	(m)	Attribute:	IP Multicast address

SERVICES:

1 (o) OpsService: Set ARP Cache

6.3.11.2.3 Attributes

Implicit
The attribute Implicit indicates that the IP suite object is implicitly addressed by the service.

IP
This attribute contains the following attributes:

IP Address
This attribute contains the current IP address according to RFC 791.

Attribute type: Unsigned32

Subnet Mask
This attribute contains the current subnet mask according to RFC 791.

Attribute type: Unsigned32

Standard Gateway
This attribute contains the current IP address of the standard gateway according to RFC 791.

Attribute type: Unsigned32

ARP
This attribute contains the following attributes:

Cache Size
This attribute contains the size of the ARP cache and shall contain a value to cache all devices that the device is connected to. The default value shall be 64 for IO devices and 256 for IO controller.

NOTE The cache size influences the performance of the IO operation.

Cache Timeout
This attribute contains the timeout value for the refresh of the ARP cache entry (see RFC 826). This attributes influences the time to recognize changed IP address MAC address couples in case of device replacement. The default value shall be 180 s.

NOTE Static ARP cache entries for IO devices provide a suitable solution if supported by the UDP/IP stack.

List of Cache Entries
This attribute list contains the following attributes:

IP Address
This attribute contains the IP address according to RFC 791.

Attribute type: Unsigned32

IEEE 802.3 MAC Address
This attribute contains the value of the ISO/IEC 8802-3 MAC address corresponding to the IP address.

Attribute type: OctetString[6]

List of IP Multicast addresses
This attribute list contains the following attributes:

IP Multicast address
This attribute contains the IP Multicast address used for multicast communication relation. It shall be set according RFC 2365 and shall follow the building rule in relation to the Frame ID shown in Table 107.

Allowed values with associated MAC address and Frame ID are listed in Table 107

Table 107 – IP Multicast address

IP multicast address	Associated Multicast MAC address	Associated Frame ID	Usage
239.192.248.0	01-00-5E-40-F8-00	0xF800	Used for multicast communication relations in conjunction with RT_CLASS_UDP
239.192.248.1 – 239.192.251.254	01-00-5E-40-F8-01 – 01-00-5E-40-FB-FE	0xF801 – 0xFBFE	Used for multicast communication relations in conjunction with RT_CLASS_UDP
239.192.251.255	01-00-5E-40-FB-FF	0xFBFF	Used for multicast communication relations in conjunction with RT_CLASS_UDP

6.3.11.3 IP suite service specification

6.3.11.3.1 Set ARP Cache

This local service may be used to manipulate the local ARP cache loading entries. Table 108 shows the parameters of the service.

Table 108 – Set ARP Cache

Parameter name	Req	Cnf
Argument	M	
Number of Entries	M	
List of Cache Entries	U	
IEEE 802.3 MAC Address	M	
IP Address	M	
Result(+)		S(=)
Result(-)		S(=)

Argument
The argument shall convey the service specific parameters of the service request.

Number of Entries
This optional parameter list shall contain the number of Cache Entries that follows. The number shall not exceed the value of the attribute Cache Size.

List of Cache Entries
The optional parameter list shall consist of the following parameter:

IEEE 802.3 MAC Address
This parameter contains the value for the equivalent ASE attribute.

IP Address
This parameter contains the value for the equivalent ASE attribute.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

6.3.12 Domain name system ASE

6.3.12.1 Overview

The utilization of the RFC 1034 (DNS) standard is defined for this specification. It includes the usage and the content syntax of domain name.

6.3.12.2 DNS class specification

6.3.12.2.1 General

The DNS ASE defines one DNS object type:

6.3.12.2.2 Template

A DNS object is described by the following template:

ASE: **DNS ASE**
CLASS: **DNS**
CLASS ID: **not used**
PARENT CLASS: **top**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	DNS
2.1	(m)	Attribute:	Primary DNS Server
2.2	(m)	Attribute:	Secondary DNS Server

SERVICES:

1	(o)	OpsService:	DNS Get Host By Name

6.3.12.2.3 Attributes

Implicit
The attribute Implicit indicates that the DNS object is implicitly addressed by the service.

DNS
This attribute contains the following attributes:

Primary DNS Server
This attribute contains the IP address of the primary DNS server according to RFC 791.

Attribute type: Unsigned32

Secondary DNS Server
This attribute contains the IP address of the secondary DNS server according to RFC 791.

Attribute type: Unsigned32

6.3.12.3 DNS service specification

The DNS services shall be derived from the RFC 1034 (DNS) standard.

6.3.13 Dynamic host configuration ASE

6.3.13.1 Overview

The utilization of the RFC 2131 (DHCP) standard is defined for this specification. It includes the usage and the content syntax of Client ID.

6.3.13.2 DHCP class specification

6.3.13.2.1 General

The DHCP ASE defines one DHCP object type:

6.3.13.2.2 Template

A DHCP object is described by the following template:

ASE: **DHCP ASE**
CLASS: **DHCP**
CLASS ID: **not used**
PARENT CLASS: **top**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit

SERVICES:

1 (o) OpsService: DHCP Get IP

6.3.13.2.3 Attributes

Implicit
The attribute Implicit indicates that the DHCP object is implicitly addressed by the service.

6.3.13.3 DHCP service specification

The DHCP services shall be derived from the RFC 2131 (DHCP) standard.

6.3.14 Simple network management ASE

6.3.14.1 Overview

The utilization of the RFC 2674 (Bridges with traffic classes), RFC 2737 (MIB 2), RFC 2863 (IF MIB), RFC 3418 (SNMP), RFC 3621 (Power over Ethernet MIB), RFC 3636 (MAU MIB) standards are defined for this specification. It includes the usage of different MIBs. Furthermore, a PNIO-LLDP MIB is specified in the protocol part.

6.3.14.2 SNMP class specification

6.3.14.2.1 General

The SNMP ASE defines one SNMP object type:

6.3.14.2.2 Template

A SNMP object is described by the following template:

ASE: **SNMP ASE**
CLASS: **SNMP**
CLASS ID: **not used**
PARENT CLASS: **top**
ATTRIBUTES:
1 (m) Key Attribute: Implicit
2 (m) Attribute: Type10 MIB
2.1 (m) Attribute: Enterprise number
2.1.1 (o) Attribute: List of Vendors
2.1.2 (m) Attribute: Vendor OUI
SERVICES:
1 (o) OpsService: SNMP Get

6.3.14.2.3 Attributes

Implicit
The attribute Implicit indicates that the SNMP object is implicitly addressed by the service.

Type10 MIB
This attribute is composed of the following elements:

Enterprise number
This attribute contains the enterprise number as MIB identifier. The value shall be according to the Table 109.

Table 109 – Enterprise number

Value	Meaning
24 686	Enterprise number for Type10 MIB

List of Vendors
This list attribute consists of the following attributes:

Vendor OUI
This attribute defines which additional vendor specific MIB extension exists.

Attribute type: Unsigned32

The allowed values are shown in Table 110.

Table 110 – Vendor OUI

Value	Meaning
Type10 OUI	Type10 defines subsequent structured data
Vendor OUI	Vendor defines subsequent structured data
Others	Reserved

6.3.14.3 SNMP service specification

The SNMP services shall be derived from the RFC 3418 (SNMP) standard.

6.3.15 Common DL mapping ASE

6.3.15.1 Overview

This ASE provides a common interface for DL mapping.

6.3.15.2 DL Mapping class specification

6.3.15.2.1 General

The DL Mapping ASE defines one DL Mapping object type:

6.3.15.2.2 Template

A DL Mapping object is described by the following template:

ASE: **DL Mapping ASE**
CLASS: **DL Mapping**
CLASS ID: **not used**
PARENT CLASS: **top**
ATTRIBUTES:
1 (m) Key Attribute: Implicit
SERVICES:
1 (m) OpsService: IRT Schedule Add
2 (m) OpsService: IRT Schedule Remove
3 (m) OpsService: Schedule
4 (m) OpsService: N Data
5 (m) OpsService: A Data
6 (m) OpsService: C Data

6.3.15.2.3 Attributes

Implicit
The attribute Implicit indicates that the DL Mapping object is implicitly addressed by the service.

6.3.15.3 DL Mapping service specification

6.3.15.3.1 IRT Schedule Add

The Schedule Add service is used to store ASE attributes by means of a local service.

Table 111 shows the parameter of the service.

Table 111 –IRT Schedule Add

Parameter name	Req	Cnf
Argument	M	
CREP	M	
D_Port	M	
Reduction Ratio	M	
Phase	M	
Result(+)		M
CREP		M
D_Port		M
Status		M

Argument
The argument shall convey the service specific parameters of the service request..

CREP
This parameter is the local identifier for the desired CR.

D_Port
This parameter contains the Port ID.

Reduction Ratio
This parameter contains the value for the equivalent attribute of the CRL class specification.

Phase
This parameter contains the value for the equivalent attribute of the CRL class specification.

Result(+)
This parameter indicates that the service request succeeded.

Status
This parameter shall contain the value OK.

6.3.15.3.2 IRT Schedule Remove

The Schedule Remove service is used to remove the schedule by means of a local service.

Table 112 shows the parameter of the service.

Table 112 – IRT Schedule Remove

Parameter name	Req	Cnf
Argument	M	
D_Port	M	
Result(+)		M
CREP		M
D_Port		M
Status		M

Argument
The argument shall convey the service specific parameters of the service request.

D_Port
This parameter contains the Port ID.

Result(+)
This parameter indicates that the service request succeeded.

CREP
This parameter is the local identifier for the desired CR.

Status
This parameter shall contain the value OK.

6.3.15.3.3 Schedule

The Schedule service is used to set the phase by means of a local service.

Table 113 shows the parameter of the service.

Table 113 –Schedule

Parameter name	Req	Cnf
Argument	M	
Phase	M	
Len	M	
Result(+)		M
Status		M

Argument
The argument shall convey the service specific parameters of the service request.

Phase
This parameter contains the phase.

Len
This parameter contains the duration of the phase.

Result(+)
This parameter indicates that the service request succeeded.

Status
This parameter shall contain the value OK.

6.3.15.3.4 N Data

This service is used to transfer non RTA and RTC data from the IO device to the IO controller and vice versa. Table 114 shows the parameters of the service.

Table 114 – N Data

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
CREP	M	M(=)	
D Port	M	M(=)	
T Stamp	M	M(=)	
DA	M	M(=)	
SA	M	M(=)	
Prio	M	M(=)	
VLAN Tag	M	M(=)	
LT	M	M(=)	
N SDU	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

D Port
This parameter contains the desired Port ID.

T Stamp
This parameter contains the time stamp.

DA
This parameter contains the destination address according to IEEE 802.3.

SA
This parameter contains the source address according to IEEE 802.3.

Prio
This parameter contains the priority according to IEEE 802.1Q.

VLAN Tag
This parameter contains the VLAN field according to IEEE 802.1Q.

LT
This parameter contains the priority according to IEEE 802.3.

N SDU
This parameter contains the APDU.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Status
This parameter contains the reason of the error.

6.3.15.3.5 A Data

This service is used to transfer RTA data from the IO device to the IO controller and vice versa. Table 115 shows the parameters of the service.

Table 115 – A Data

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
CREP	M	M(=)	
D Port	M	M(=)	
T Stamp	M	M(=)	
DA	M	M(=)	
SA	M	M(=)	
Prio	M	M(=)	
VLAN Tag	M	M(=)	
LT	M	M(=)	
A SDU	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

D Port
This parameter contains the desired Port ID.

T Stamp
This parameter contains the time stamp.

DA
This parameter contains the destination address according to IEEE 802.3.

SA
This parameter contains the source address according to IEEE 802.3.

Prio
This parameter contains the priority according to IEEE 802.1Q.

VLAN Tag
This parameter contains the VLAN field according to IEEE 802.1Q.

LT
This parameter contains the priority according to IEEE 802.3.

A SDU
This parameter contains the APDU.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Status

This parameter contains the reason of the error.

6.3.15.3.6 C Data

This service is used to transfer RTC data from the IO device to the IO controller and vice versa. Table 116 shows the parameters of the service.

Table 116 – C Data

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
CREP	M	M(=)	
D Port	M	M(=)	
DA	M	M(=)	
SA	M	M(=)	
Prio	M	M(=)	
VLAN Tag	M	M(=)	
LT	M	M(=)	
N SDU	M	M(=)	
APDU Status	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument
The argument shall convey the service specific parameters of the service request.

CREP
This parameter is the local identifier for the desired CR.

D Port
This parameter contains the desired Port ID.

DA
This parameter contains the destination address according to IEEE 802.3.

SA
This parameter contains the source address according to IEEE 802.3.

Prio
This parameter contains the priority according to IEEE 802.1Q.

VLAN Tag
This parameter contains the VLAN field according to IEEE 802.1Q.

LT
This parameter contains the priority according to IEEE 802.3.

C SDU
This parameter contains the APDU.

APDU Status
This parameter contains the field APDU Status.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Status
This parameter contains the reason of the error.

7 Communication model for distributed automation

7.1 Concepts

7.1.1 Overview

The distributed automation specification provides an advanced FAL object model necessary for distributed automation between smart automation devices having equal rights and having a facet for the remote engineering of the system. These two facets are well defined within the distributed automation FAL application entity.

Subclause 7.1 introduces the basic concept of the distributed automation FAL application entity (FAL AE). The distributed automation FAL is composed of distributed automation specific APO ASEs and one AR ASE. The APO ASEs provide different services and attributes which are structured into groups referred to as interfaces. Each interface provides a certain view on the object instance and is uniquely referenced via a local address or a reference referred to as Interface Pointer. The distributed automation APO ASEs have a well defined relationship among each other – also within a device- referred to as Runtime Object Model. This distributed automation basic model is described in 7.1.5.

The distributed automation FAL AE maps the distributed automation ASEs via one AR ASE onto a set of underlying services and protocols modeled as an abstract ORPC model. These services are referred to as ORPC services and the protocol is referred to as ORPC wire protocol. Any existing and future ORPC services and protocols (e.g. in form of specific middleware) that can be mapped to the defined abstract services can be used. The abstract ORPC model provides services and protocols that may create an object instance from an globally unique identified object class or interface from that class, provides and manages all binding information to the remote object instance by means of Interface Pointers, provides at least one confirmed service to convey defined distributed automation ASE services, and finally provides data marshaling for all service parameters by means of a well defined interface description referred to as interface definition language (IDL).

7.1.2 Objectives of the FAL AE

One objective of the distributed automation FAL AE is to provide services for distributed automation by means of ORPC services. This allows distributed automation devices (e.g. controller, drives, field devices, I/O devices, HMI devices) to be connected and exchange data values both as consumer and/or provider in an optimized way.

Another objective of the distributed automation FAL AE is to provide services for the engineering of distributed automation devices and distributed automation functions by means of ORPC services. This allows one or more engineering devices to configure an automation

system via interconnecting data items as source or sink and invoke services to save that connection information persistent within each consumer part of an automation device. Furthermore, engineering clients may use services to read diagnosis information at different levels from automation devices. The engineering is also supported by a well defined navigation procedure through the linked object instances of the APO ASE classes with the Physical Device Class as the navigation anchor.

Another objective is the seamless business integration. The distributed automation FAL ASE in combination with the chosen ORPC model is able to communicate seamlessly with common office application e.g. as HMI tools.

Another objective the distributed automation FAL AE is to connect distributed automation devices with type 3 devices by means of a proxy device. Therefore, distributed applications are supported on devices that are connected via different networks.

In summary, the distributed automation FAL AE enables:

– distributed automation systems to be constructed and engineered from distributed automation devices,
– remote applications (e.g. enterprise resource planning systems, office applications) may access field devices or controller of any type through ORPC services using the distributed automation communication model,
– engineering applications to interconnect data items in a common way,
– type 3 or other networks to be linked via a distributed automation proxy device.

7.1.3 Devices types

Distributed automation FAL devices are FAL devices that contain a distributed automation FAL AE with exactly one instance of the Physical Device Class (PDev), with one or more instances of the Logical Device Class (LDev), with exactly one instance of the Active Connection Control Object Class (ACCO) per LDev instance, and with one or more instances of the RT-Auto Class per LDev instance. These distributed automation FAL devices are also referred to as automation devices.

Another class of distributed automation FAL devices is capable to manage the connection of data items of instances of RT-Auto classes, read diagnosis information from automation devices, and in summary is capable to use the services from the FAL objects as a client. This type of FAL device is referred to as engineering device.

Another class of distributed automation FAL devices is capable to connect a type 3 network or other networks to a distributed automation network and to provide the distributed automation object model as a proxy for type 3 or other devices. This type of FAL device is referred to as proxy device.

A device manufacturer may also implement different distributed automation FAL device classes inside one product as mixed implementation.

7.1.4 FAL ASEs

Figure 13 shows the distributed automation FAL ASEs from the communication perspective. The application process uses ASE service primitives to influence remote objects. The services are grouped by means of interfaces. The only address information is the Interface Pointer. The Interface Pointer contains implicit binding information (e.g. IP address and reference to an interface on a particular object instance on that node). These services are all mapped to the AR-Call service. The AR-CoCreateInstance service is directly called by the application process to achieve a binding to a remote PDev. The AR ASE conveys the services by means of the services provided by the ORPC model. In addition to the unified service parameter the AR ASE adds a reference to the IDL description. This reference is necessary for the

marshalling of service parameter performed by the ORPC sublayer. Eventually, the application layer protocol data unit (APDU) is build by the ORPC wire protocol and conveyed to the remote host.

NOTE This communication procedure may also be used for local communication within the device as a local matter.

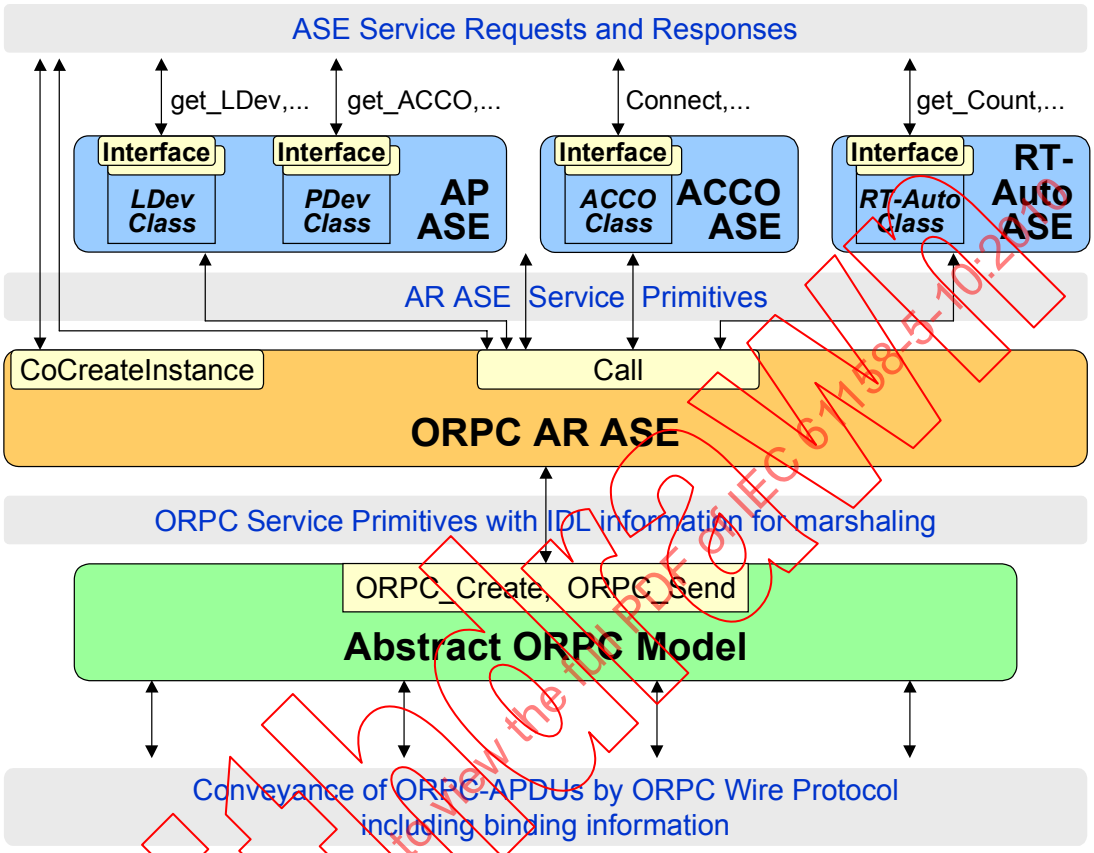


Figure 13 – FAL ASEs communication architecture

The relationships and linking between the FAL ASEs is provided by the Runtime Object Model that is defined in the subclause below.

7.1.5 Runtime object model

The distributed automation runtime object model represents the objects that exist in a device, together with their interfaces grouping services and attributes. The model also describes the interrelationship between the individual objects.

The Figure 14 specifies the runtime object model of distributed automation.

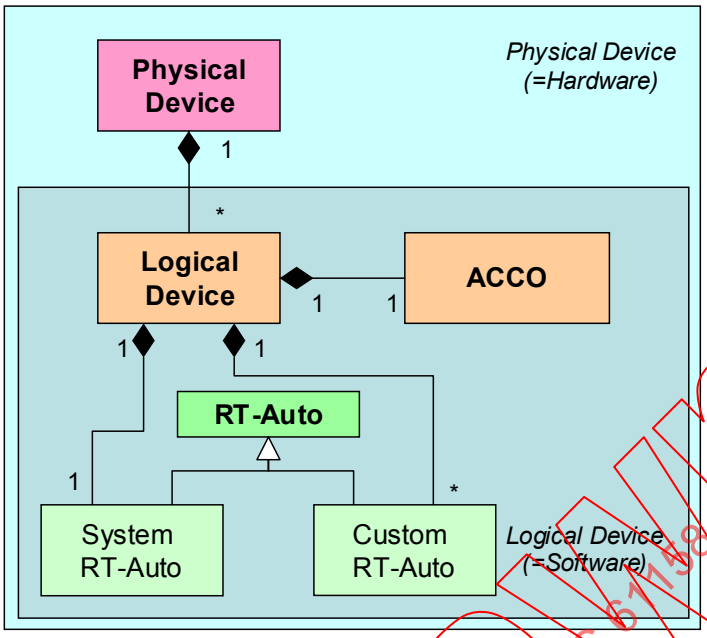


Figure 14 – Runtime object model

In the runtime object model, the Physical Device (PDev) represents a physical component (i.e. hardware). It offers a network access to one or more IP networks. The PDev exposes the physical properties of the component via the ICBAPhysicalDevice and ICBAPhysicalDevice 2 interfaces. Exactly one instance of the PDev exists in each hardware component (e.g. PLC, drive, PC).

A PDev references one or more Logical Devices (LDev). An LDev represents a software component or a firmware component as an autonomous self-contained unit. The LDev represents the functions of a device for example an actuator, sensor, controller or CNC. The devices solve the distributed automation task.

The PDev is able to navigate to all contained LDev by their names. It offers browsing functionality for the names of the LDevs.

The PDev shall have the distributed automation ClassID (defined in 7.3.2.1.2) that is the same for all devices participating in distributed automation fieldbus. A communication partner without any knowledge of the destination hardware/software can obtain an Interface Pointer to the PDev object. This object is used for navigating to any further object.

In most cases, there is a firm one to one allocation between PDev and LDev objects. This means that there is exactly one firmware related to a hardware component. Examples are PLC, drives or field devices.

In addition, a one to n allocation is possible. It is used for devices that offer several independent functions in a hardware. Examples are free computing power within a PC with SlotPLC and SoftPLC, or a PC with HMI and PLC functionality.

The LDev comprises the components operating system and application. The operating system consists of two parts: A standardized part and an enhanced part that is used for special definitions of the LDev programmer.

The distributed automation specification uses the Logical Device and ACCO class to describe the functionality that is the same throughout all LDevs.

The Logical Device class provides general automation functions such as identification or diagnosis, and is used as a navigation anchor for all other objects of the operating system. In addition, it is able to navigate to all referenced RT-Auto objects by their names and it offers browsing functionality for the names of the RT-Auto objects.

The ACCO (Active Control Connection Object) class provides a configurable connection of RT-Auto objects.

For both, functionality that is specified by distributed automation is required. In the following, this functionality is marked as mandatory or optional. Mandatory interfaces shall be offered, optional interfaces may be offered. If a functionality is offered that is specified by an optional interface, this shall be done using the described interface.

The LDev and ACCO may provide manufacturer specific functions that go beyond distributed automation specification. This functionality may be defined by a manufacturer and/or a trade organization. This enhanced functionality shall be expressed in form of interfaces. It shall be defined by an interface definition in the form of an IDL. The QueryInterface mechanism supports inquiry whether a specific device supports optional functionality.

The RT-Auto ASE represents the automation functionality in the form of a process-related component. Examples are boiler, controller, etc. These automation functionalities of an application may exist in the form of pre-tested, self-contained and universally useable components. Using the interconnecting functionality that is defined by ACCO ASE, automation functionalities can be configured as a distributed application.

System RT-Auto objects allow access to system variables. Currently one System RT-Auto object ("!SYSTEM") is defined to allow connections being established with specific variables like State or GroupError.

The Figure 15 specifies the relationship between engineering and runtime.

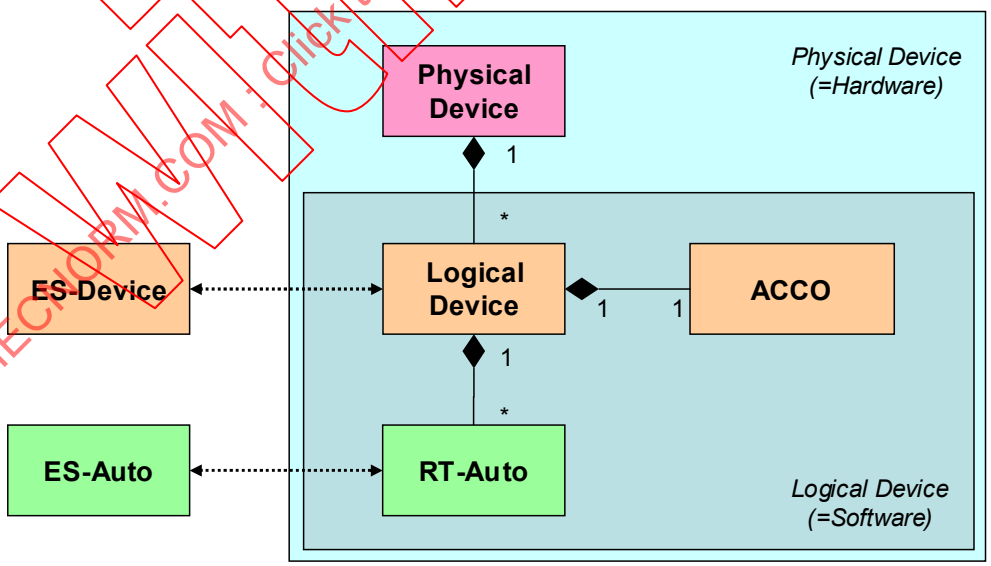


Figure 15 – Relationship between engineering and runtime

Each RT-Auto object has a representative in engineering. All the other objects of the LDev together have one engineering system device as representative in the engineering. This representative hides the internal object structure and the producer-specific features of the enhanced LDev towards engineering.

The following basic rule applies to separating a device of fixed functionality into RT-Auto and LDev objects: If the function of a LDev object shall be interconnectable in an application, this function shall be implemented via an RT-Auto object of a fixed functionality. The producer-specific properties of the LDev chiefly contain the functionality that is used for hardware configuration, debugging and other similar services. The properties of the LDev can only be used by the LDev's engineering representative.

The LDev shall have a functioning default so that it can be used immediately after installation in a basic scope and without parameter value assignment. In particular, this concerns definitions with regard to a default configuration for the IP suite ASE.

7.1.6 Navigation in the object model

Two different procedures may be used for navigating to an object of the runtime object model:

– The navigation to the PDev of a distributed automation device is performed via IP address using the ORPC AR service CoCreateInstance (see 7.4.3.1).

– The navigation within the runtime object model takes place via attributes (properties) at the individual objects (as shown in Figure 16).

– If these are one to n relations, the navigation between PDev and LDev, and between LDev and RT-Auto objects are based on names. PDev offers the LogicalDevice property and LDev offers the RTAuto property for this purpose.

– Each of the LDev and RT-Auto objects offers the Name property for self-information.

The tag and the parent property that are available at the PDev and LDev objects shall solely be used for supporting identification systems, it is not used for navigation.

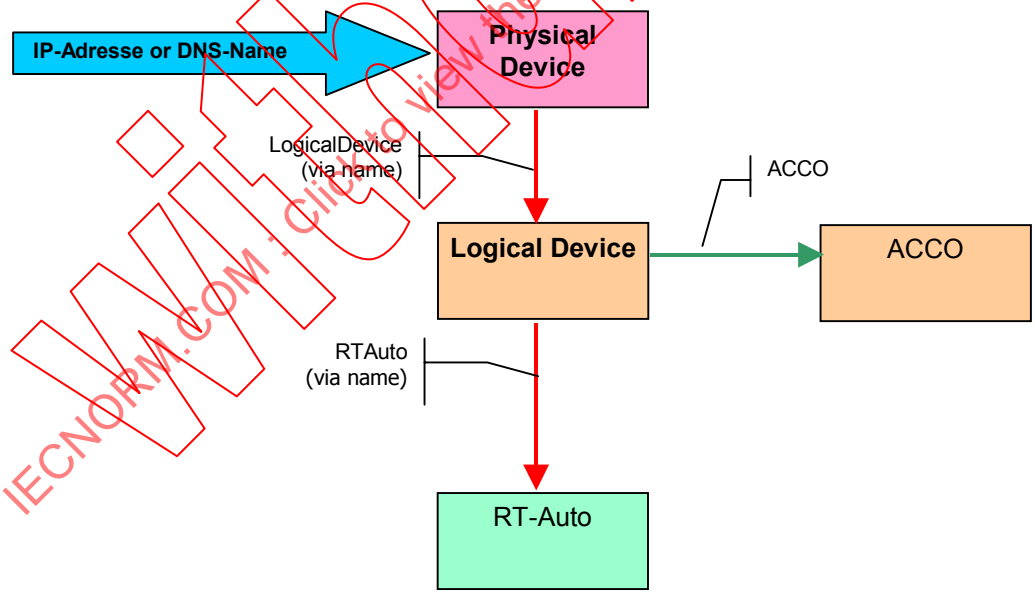


Figure 16 – Navigation in the runtime object model

7.1.7 Online/offline comparison

The objective of the on-line / off-line comparison is to verify whether off-line project (i.e. the one found in the engineering system) and on-line project (i.e. the one currently running in the runtime) agree with respect to their

- object world (i.e. the versions with respect to the RT-Autos, LDev), and their
- connections (connection adjustment).

A given project is being stamped in several ways:

(1) PDev Stamp
This stamp will alter, if one of its associated LDev objects is being removed, a new one is being added or an LDev objects name changed. For detailed description see 7.3.2.1.3.2.

(2) ACCO Stamp
This stamp will alter, if connections are added, removed, or if one of the existent connections or the ping-factor has been changed. For detailed description see 7.3.3.3.4.12.

All stamps are being issued by runtime only, the engineering system will never issue any stamps, since it cannot provide for uniqueness and integrity since changes could be conducted outside the engineering system, either by a different one or another engineering tool.

NOTE Priority for stamping is to reliably indicate difference rather than indicate equity! This means particularly that a stamp does not necessarily return to equity if changes get undone.

7.2 ASE data types

7.2.1 Supported data types

The data types supported by distributed automation are a subset of those defined in clause 5. They are:

- ACCESSRIGHTSDEF
- ADDCONNECTIONIN
- ADDCONNECTIONOUT
- BSTR
- char
- CONNECTIN
- CONNECTIN2
- CONNECTINCR
- CONNECTINSRT
- CONNECTOUT
- CONNECTOUTCR
- date
- DIAGCONSCONNOUT
- DISPPARAMS
- double
- EXCEPINFO
- HRESULT
- FILETIME
- float
- GETCONNECTIONOUT
- GETCONSCONNOUT
- GETIDOUT
- GETPROVCONNOUT
- GROUPERRORDEF
- HRESULT
- Interface Pointer

- ITEMQUALITYDEF
- long
- LPWSTR
- MACAddr
- OctetString
- PERSISTDEF
- READITEMOUT
- SAFEARRAY
- short
- STATEDEF
- struct (see Note below)
- unsigned char
- unsigned long
- unsigned short
- UUID
- VARIANT
- VARTYPE
- VARIANT_BOOL
- WRITEITEMIN
- WRITEITEMQCDIN

NOTE The term struct is used to address user defined structured data types that should be built according to IEC/TR 61158–1:2010 using the base data types that are listed in this subclause.

7.2.2 Connectable data types

Table 117 defines the data types that are valid for data connections. They define a subset of the the distributed automation supported data types that is expedient for automation.

Table 117 – Connectable data types

VARTYPE	Data type Name
VT_BOOL	VARIANT_BOOL
VT_I1	char
VT_UI1	unsigned char
VT_I2	short
VT_UI2	unsigned short
VT_I4	long
VT_UI4	unsigned long
VT_R4	float
VT_R8	double
VT_DATE	date
VT_BSTR	BSTR
VT_SAFEARRAY	SAFEARRAY
VT_USERDEFINED	struct

Only data of the same data type can be connected with each other. For an array, in particular, this means dimension, length and base data type of source and destination shall be compatible with each other. Within a structure, the layout of source and destination shall be compatible (i.e. the data types of the members).

NOTE A SAFEARRAY with base type VT_BSTR is only connectable if both connection source and connection sink have the same maximum length. The same applies to a structure with elements of type VT_BSTR; it will only

be connectable if every BSTR element of the structure has the same maximum length on both provider and consumer.

The maximum size of a property is 32 kByte in its representation according to the standard format for connection data.

Userdefined structures are restricted to a nesting depth of 1, i.e. a structure cannot itself have an element of type structure or SAFEARRAY. The maximum dimension of SAFEARRAYS is restriced to 1, i.e. only one-dimensional SAFEARRAYs are supported. The element data types and base data types are therefore restricted to simple data types.

For the data type date only values greater than “01.01.1900 00:00” are supported.

Table 118 shows the supported data types according to the Base Object Version.

Table 118 – Supported data types according to the Base Object Version

VARTYPE	standalone	in array VT_SAFEARRAY	in structure VT_USERDEFINED	Restrictions
VT_BOOL	1	2	2	—
VT_I1	1	1	2	—
VT_UI1	1	1	2	—
VT_I2	1	1	2	—
VT_UI2	1	1	2	—
VT_I4	1	1	2	—
VT_UI4	1	1	2	—
VT_R4	1	2	2	—
VT_R8	2	2	2	—
VT_DATE	1	2	2	≥ 01.01.1900 00:00
VT_BSTR	1	2	2	—
VT_SAFEARRAY	1	X	X	One-dimensional only
VT_USERDEFINED	2	X	X	maximum nesting depth 1
X means not supported. 1 means supported with Base Object Version 1 and greater. 2 means supported with Base Object Version 2 and greater.				

7.2.3 Extended type description

The extended type description describes unambiguous types, whether simple (scalar) or complex (string, structure or array), and is used to transport type descriptions together with data. It is represented by an ARRAY of WORD.

The extended type description is build according the following rules:

- Scalar types (VT_BOOL, VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8 and VT_DATE) are described through a single word, their according VARTYPE.
- VT_BSTR is described by two words: the type (VT_BSTR), followed by a word holding the maximum length in bytes without the terminating '\0', according to the definition of the BSTR length.
- VT_USERDEFINED is described by multiple words: the type (VT_USERDEFINED), followed by a word holding the count of elements, then followed by the type description of each structure element.
- The type description of the structure elements is build depth first, meaning if an element is itself again a structure, the elements of the substructure will first be placed in the overall type description, then followed by the elements of the structure.
- VT_ARRAY is described by multiple words: the type (VT_ARRAY), followed by a word holding the dimension, followed by "dimension" words holding the number of elements per dimension, followed by the type description of the array's base type.

The type description is always complete with itself, e.g. there are no references to subtypes, even if they are used frequently. The names of types and subtypes are not held in runtime (for example names of structure elements).

An example for an extended type description follows below:

Pseudo-code Declaration	type Description	Comment
ARRAY[2,3] of {	VT_ARRAY	Array
	2	Dimension 2
	2	Length 2
	3	Length 3
	VT_USERDEFINED	of struct
	4	with 4 elements
i4;	VT_I4	Element 0: long
BSTR[120];	VT_BSTR	Element 1: BSTR
	120	maxlen 120
{	VT_USERDEFINED	Element 2: struct
	2	with 2 elements
date;	VT_DATE	Element 0: DATE
ARRAY[10] von i4;	VT_ARRAY	Element 1: Array
	1	Dimension 1
	10	Length 10
	VT_I4	of long
}		
ui4;	VT_UI4	Element 3: unsigned long
}		

7.2.4 Definitions for identifiers

7.2.4.1 Character set type 1

The character set type 1 is defined as a subset of the ISO 8859-1 character set (ISO Latin-1 Character Set). This character set permits a West-European notation of identifiers. Thus, at least the characters of languages like English, German, Italian, Spanish or French can be used. One byte is used for storing the characters. Supporting more advanced character sets (such as DBCS or Unicode for Kanji or something similar) is disapproved since this would mean the introduction of locale information in runtime.

To permit all variants with respect to the nomenclature to be used, all „printable" characters from the ISO 8859-1 character set may be used. The following characters shall not be used:

- characters with the code 0 – 31
- characters with the code 127 – 159
- the character with code 33 ("!") is restricted for internal use.

These definitions of the character set permit identifiers to be stored in a compressed format. This means that Wide Character Strings need not be stored; these can be reconstructed from a normal character string by just adding a zero for the second byte of each wide character.

7.2.4.2 Character set type 2

The character set type 2 is defined as a subset of the ISO 10646-1 character set (also known as Universal multiple-octet coded Character Set (UCS) Row 00, or as ASCII). The character set spans the „lower half" of the ISO 8859-1 character set, i.e. the characters with code 0 – 127. The character set is further restricted to the following characters:

- letters (A-Z a-z); characters with the code 65-90, resp. 97-122
- figures (0-9); characters with the code 48-57
- underscore („_"); character with the code 95

7.2.4.3 Common definitions

Object names (for LDev and RT-Auto objects), attribute names and service names are considered as identifiers. Identifiers shall be stored in a case-preserving way, but the comparison of all identifiers shall be case insensitive.

NOTE A case-insensitive comparison for character set type 1 (and type 2 where applicable) can easily be implemented since capital and small letters are mapped in parallel code areas. The characters of the following character strings are converted into the corresponding capital letters by merely subtracting 32: 97 (a) – 122 (z), 224 (à) – 246 (ö), 248 (ø) – 254 (þ). All the other characters are retained in their original form.

At the interface level Wide Character Strings are transported by either using the data type BSTR or the data type LPWSTR, whereas BSTR has a length field permitting NULL characters in its wide character string, LPWSTR is a pointer to a null-terminated wide character string. BSTR is used on all interfaces derived from the Dispatch interface (on LDev or RT-Auto), LPWSTR on all interfaces derived from the Unknown interface (on the ACCO).

The maximum length of identifiers is defined in order to permit an optimum implementation of the persistence of identifiers in the ACCO ASE context. A maximum length of 32 usable characters is defined for object names, tag names and method names.

7.2.4.4 Usage of character sets

Table 119 defines the usage of the character sets.

Table 119 – Usage of character sets

Identifier	Character set	Restrictions
PDev object name	4 dotted decimal	
LDev object name	character set type 1	maximum length of 32 characters
RT-Auto object name	character set type 1	maximum length of 32 characters
attribute name, service name or event name	character set type 2	maximum length of 32, shall begin with a letter

7.3 ASEs

7.3.1 Object Management ASE

7.3.1.1 Overview

The Object Management ASEs provides a common set of management services applicable to all classes of objects, except for the Data type classes. This management services operate on FAL APOs and their attributes. FAL APO classes are defined within the FAL ASEs that are specified in the remaining clauses. Each FAL APO class defined is described by a set of attributes and a set of services.

Within the distributed automation FAL ASEs the term interface is used to structure attributes and services into different views of a FAL Class. This interface or attribute structure of a class is uniquely identified by a globally unique identifier (UUID) referred to as Interface ID. The parameter Interface Pointer contains the local address of the interface itself during runtime and defines an unambiguous reference. The reference is volatile and the value shall be always obtained online by means of specific distributed automation FAL services. It is only valid (e.g. indicated by specific HRESULT values) until the server fails in the sense of communication (e.g. power down, communication failure).The reference stays invalid, even if the server hosting the object with former referenced interface is up and running again. Therefore, the client gets a valid Interface Pointer reestablishing the Interface Pointer anew. There are three possibilities to do this. The Interface Pointer is returned by the object instance of the PDev Class accessed via the AR service CoCreateInstance using IP address, ClassID and Interface ID. A second alternative is to ask the Unknown interface of an object for another interface of the same class via the management service QueryInterface. A third alternative is to use specific services (e.g. get_LogicalDevice) to navigate through the objects.

The distributed automation FAL services are utilizing an abstract object remote procedure call (ORPC) model that defines exactly one AREP for the FAL services. Therefore, the local addressing of a FAL service is implicit and the service parameter AREP is omitted. The conveyance of the FAL service parameter, network addressing and, e.g. managing of network connections is part of the underlying ORPC wire protocol and goes beyond the FAL service specification.

The Object Management ASE provides two groups of object management attributes and services: the Unknown interface and the Dispatch interface.

The Unknown interface is the fundamental interface and mandatory for each distributed automation FAL class. It supports resource management (AddRef and Release), type coercion (QueryInterface) and identity services. The Unknown interface is the root of all interfaces. This means that it is not derived from any further interface. All other interfaces are derived either directly from the Unknown or from another interface that can be in turn derived from the Unknown interface. The Unknown interface contains exactly three services: QueryInterface, AddRef and Release. Since each interface needs to eventually be derived from the Unknown interface, these methods can be invoked from any interface of an object. The Unknown interface itself is a separate interface with its own interface identifier. Thus, a client can obtain an Interface Pointer via QueryInterface that points specifically to the Unknown interface.

The Dispatch interface is a second interface supported by the Physical Device, Logical Device and RT-Auto classes. It is designed in such a way that it can call virtually any other FAL class interfaces if supported by the particular class. Distributed automation FAL classes supporting the Dispatch interface shall contain all attributes and all services of the object instance.

NOTE 1 The Dispatch interface can be used for the integration into script languages. Essentially, it permits a symbolic access to a distributed automation FAL object to be made.

NOTE 2 Scripts use only one access to distributed automation FAL objects – via the Dispatch interface that is returned by the QueryInterface service. To enable scripts to access all attributes and services of an object, the implementation of the Dispatch interface contains all members of all interfaces of the object that have been derived from the Dispatch interface. In a way, the Dispatch interface aggregates all attributes and services of all interfaces to one "union set".

7.3.1.2 Class specification

7.3.1.2.1 Template

FAL ASE:	**Object Management ASE**
CLASS:	**Base Object**
CLASS ID:	**not used**
PARENT CLASS:	**TOP**
ATTRIBUTES and SERVICES:	
1	(m) Key Attribute: implicit

2	(m)	Attribute:	Base Object Version
2.1	(m)	Attribute:	Major
2.2	(m)	Attribute:	Minor
3	(m)	Attribute:	Interface Unknown
3.1	(m)	MgtService:	QueryInterface
3.2	(m)	MgtService:	AddRef
3.3	(m)	MgtService:	Release
4	(m)	Attribute:	Interface Dispatch
4.1	(m)	MgtService:	GetTypeInfoCount
4.2	(m)	MgtService:	GetTypeInfo
4.3	(m)	MgtService:	GetIDsOfNames
4.4	(m)	MgtService:	Invoke

7.3.1.2.2 Attributes

Base Object Version
This attribute specifies the base object version of all ASEs.

Major
This attribute contains the major part of the base objects version.

Attribute type: short

Minor
This attribute contains the minor part of the base objects version.

Attribute type: short

Interface Unknown
This attribute contains the Unknown interface of the object. It exists in one variant named IUnknown.

Interface Dispatch
This attribute contains the Dispatch interface of the object. It exists in one variant named IDispatch.

7.3.1.3 Service specification

7.3.1.3.1 Interface Unknown

7.3.1.3.1.1 QueryInterface

The QueryInterface service reads the Interface Pointer (address) of a requested interface of an object instance. The arguments of the service are shown in Table 120.

Table 120 – QueryInterface (Unknown interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Interface ID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
ppvObject			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the IUnknown interface of the requested object instance.

Parameter type: Interface Pointer

Interface ID
This parameter contains the unique interface identifier of the requested interface.

Parameter type: UUID

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the IUnknown interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

ppvObject
This parameter contains the address of the requested interface of the object instance.

Parameter type: Interface Pointer

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the IUnknown interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_NOINTERFACE

7.3.1.3.1.2 AddRef

The AddRef service increments and reads the reference count for a requested interface of an object. The reference count is a local attribute that can be used to reflect the using of an object interface. It does not have any influence on the lifetime of the object.

The arguments of the service are shown in Table 121.

Table 121 – AddRef (Unknown interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result			M	M(=)
Interface Pointer			M	M(=)
Reference Count			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the IUnknown interface of the requested object instance.

Parameter type: Interface Pointer

Result:
This parameter indicates that the service request was executed.

Interface Pointer
This parameter contains the address of the IUnknown interface of the requested object instance.

Parameter type: Interface Pointer

Reference Count
This parameter contains the new reference count.

Parameter type: unsigned long

Allowed values: any value different than zero

7.3.1.3.1.3 Release

The Release service decrements and reads the reference count for an requested interface on an object. The reference count is a local attribute that can be used to reflect the using of an object interface. It does not have any influence on the lifetime of the object. The arguments of the service are shown in Table 122.

Table 122 – Release (Unknown interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result			M	M(=)
Interface Pointer			M	M(=)
Reference Count			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the IUnknown interface of the requested object instance.

Parameter type: Interface Pointer

Result:
This parameter indicates that the service request was executed.

Interface Pointer
This parameter contains the address of the IUnknown interface of the requested object instance.

Parameter type: Interface Pointer

Reference Count
This parameter contains the new reference count.

Parameter type: unsigned long

Allowed values: any value different than zero

7.3.1.3.2 Interface Dispatch

7.3.1.3.2.1 GetTypeInfoCount

The GetTypeInfoCount service determines whether type information is available for an object or not. The arguments of the service are shown in Table 123.

Table 123 – GetTypeInfoCount (Dispatch interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pcTInfo			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pcTInfo
This parameter contains wether this object provides type information or not.

Parameter type: unsigned short

Allowed values: 0 (the object does not provide type information), 1 (the object provides type information)

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_NOTIMPL

7.3.1.3.2.2 GetTypeInfo

The GetTypeInfo service retrieves the type information for an object, which can then be used to get the type information for an interface.

NOTE The type information is derived from the deployed ORPC model and beyond the scope of the FAL service specification. It contains the description of attributes and services in order to support a late binding.

The arguments of the service are shown in Table 124.

Table 124 – GetTypeInfo (Dispatch interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
iTInfo	M	M(=)		
lcid	U	U(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
ppTInfo			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

iTInfo
This parameter contains the type information to return. Zero indicates the IDispatch interface.

Parameter type: unsigned short

Allowed values: 0

lcid
This parameter contains a language identifier for the type information. An object may be able to return different type information for different languages.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

ppTInfo
This parameter contains the requested type information.

Parameter type: ITypeInfo

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: DISP_E_BADINDEX, TYPE_E_ELEMENTNOTFOUND

7.3.1.3.2.3 GetIDsOfNames

The GetIDsOfNames service maps a single attribute name or service name and an optional set of parameter names to a corresponding set of integer dispatch identifiers, which are used on subsequent calls of the Invoke service. The arguments of the service are shown in Table 125.

Table 125 – GetIDsOfNames (Dispatch interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
riid	M	M(=)		
List of rgszNames	M	M(=)		
cNames	M	M(=)		
lcid	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of rgDispId			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

riid
This parameter is reserved and shall contain UUID_NULL.

Parameter type: UUID

List of rgszNames
This parameter refers the names to be mapped.

Parameter type: LPWSTR

cNames
This parameter contains the number of the names to be mapped.

Parameter type: unsigned short

lcid
This parameter contains a language identifier to indicate the context in which the names are to be interpreted. An object may be able to provide different languages.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

List of rgDispId
This parameter contains the requested identifiers (ID) corresponding to the names passed in the List of rgszNames. The first identifier represents the attribute name respective service name. The subsequent identifiers represent the service parameter names.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, DISP_E_UNKNOWNNAME, DISP_E_UNKNOWNLCID

7.3.1.3.2.4 Invoke

The Invoke service provides access to attributes and calls services exposed by an object. The arguments of the service are shown in Table 126.

Table 126 – Invoke (Dispatch interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
dispIdMember	M	M(=)		
riid	M	M(=)		
lcid	M	M(=)		
wFlags	M	M(=)		
pDispParams	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVarResult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pExcepInfo			C	C (=)
puArgErr			C	C (=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

dispIdMember
This parameter contains the identifier of the attribute or service to invoke.

Parameter type: long

riid
This parameter is reserved and shall contain UUID_NULL.

Parameter type: UUID

lcid
This parameter contains a language identifier to indicate the context in which the names are to be interpreted. An object may be able to provide different languages.

Parameter type: unsigned long

wFlags
This parameter contains the context of the invoke service.

Parameter type: unsigned short

Allowed values: DISPATCH_METHOD, DISPATCH_PROPERTYGET,

DISPATCH_PROPERTYPUT, DISPATCH_PROPRTYPUTREF

pDispParams
This parameter contains an array of arguments, an array of dispatch identifiers for named arguments and the number of elements in these arrays.

Parameter type: DISPPARAMS

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVarResult
This parameter contains the result of the invoked attribute or service.

Parameter type: VARIANT

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the IDispatch interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: DISP_E_BADPARAMCOUNT, DISP_E_BADVARTYPE, DISP_E_EXCEPTION, DISP_E_MEMBERNOTFOUND, DISP_E_NONAMEDARGS, DISP_E_OVERFLOW, DISP_E_PARAMNOTFOUND, DISP_E_TYPEMISMATCH, DISP_E_UNKNOWNINTERFACE, DISP_E_UNKNOWNLCID, DISP_E_PARAMNOTOPTIONAL

pExcepInfo
This parameter is only valid for hresult=DISP_E_EXCEPTION and contains the exception information.

Parameter type: EXCEPINFO

puArgErr
This parameter is only valid for hresult=DISP_E_TYPEMISMATCH or hresult= DISP_E_PARAMNOTFOUND and contains the index of the first argument containing an error.

Parameter type: unsigned short

7.3.2 AP ASE

7.3.2.1 Physical device

7.3.2.1.1 Overview

In the runtime object model, the Physical device (PDev) represents a physical component (i.e. hardware). It offers a network access to one or more IP networks. The PDev exposes the physical properties of the component via the ICBAPhysicalDevice interface. Exactly one instance of the PDev exists in each distributed automation device (e.g. PLC, drive, PC).

7.3.2.1.2 Physical Device class specification

7.3.2.1.2.1 Template

The Physical Device object is described by the following template:

FAL ASE:	**AP ASE**		
CLASS:	**Physical Device**		
CLASS ID:	**CBA00000-6C97-11D1-8271-00A02442DF7D**		
PARENT CLASS:	**Base Object**		
ATTRIBUTES and SERVICES:			
1	(m)	Key Attribute:	implicit
2	(m)	Attribute:	Interface Physical Device
2.1	(m)	Attribute:	Producer
2.2	(m)	OpsService:	get_Producer
2.3	(m)	Attribute:	Product
2.4	(m)	OpsService:	get_Product
2.5	(m)	Attribute:	SerialNo
2.6	(m)	OpsService:	get_SerialNo
2.7	(m)	Attribute:	ProductionDate
2.8	(m)	OpsService:	get_ProductionDate
2.9	(m)	OpsService:	Revision
2.10	(m)	Attribute:	List of LogicalDevice
2.10.1	(m)	Attribute:	Name
2.10.2	(m)	Attribute:	Interface Pointer
2.11	(m)	OpsService:	get_LogicalDevice
2.12	(c)	Constraint:	Base Object Version.Major >= 2
2.12.1	(m)	OpsService:	Type
2.12.2	(m)	OpsService:	PROFInetRevision
2.12.3	(m)	Attribute:	PDevStamp
2.12.4	(m)	OpsService:	get_PDevStamp
3	(m)	Attribute:	Interface Browse
3.1	(m)	Attribute:	Count
3.2	(m)	OpsService:	get_Count
3.3	(m)	OpsService:	BrowseItems
3.4	(c)	Constraint:	Base Object Version.Major >= 2
3.4.1	(m)	Attribute:	Count2
3.4.2	(m)	OpsService:	get_Count2
3.4.3	(m)	OpsService:	BrowseItems2
4	(m)	Attribute:	Interface Persist
4.1	(m)	OpsService:	Save
4.2	(c)	Constraint:	Base Object Version.Major >= 2
4.2.1	(m)	OpsService:	Save2

5 (m) Attribute: Dispatchable

7.3.2.1.2.2 Attributes

Interface Physical Device
This attribute contains the physical device interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBAPhysicalDevice includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBAPhysicalDevice2 is derived from the variant ICBAPhysicalDevice and includes such the attributes and services of ICBAPhysicalDevice and additional attributes and services for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

Producer
This attribute contains the name of the device producer.

Attribute type: BSTR (maximum length of 128 usable characters)

Product
This attribute contains the name of the device type in a producer-specific and producer-unambiguous way.

Attribute type: BSTR (maximum length of 128 usable characters)

SerialNo
This attribute contains the serial number of the Physical device.

Attribute type: VARIANT

ProductionDate
This attribute contains the production date of the Physical device.

Attribute type: date

List of LogicalDevice
This attribute contains a list of references to all related Logical Device objects. A single reference can be read via the service get_LogicalDevice. It is used for navigating in the object model. The attribute is composed of the following list elements.

Name
This attribute contains the name of the Logical Device object.

Attribute type: BSTR

Interface Pointer
This attribute contains the local address of the ICBALogicalDevice interface of the Logical Device object.

Attribute type: Interface Pointer

PDevStamp
This attribute contains a hash value over the associated LDev objects. It shall alter, if one of it is being removed, a new one is being added or a LDev objects name has changed (see 7.3.2.1.3.2 for details).

Attribute type: long

Interface Browse
This attribute contains the Browse interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBABrowse includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBABrowse2 is derived from the variant ICBABrowse and includes such the attributes and services of ICBABrowse and additional attributes and services for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

Count
This attribute contains the number of elements in the Browse database.

Attribute type: long

Count2
This attribute contains the number of elements in the Browse database.

Attribute type: long

Interface Persist
This attribute contains the persistence interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBAPersist includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBAPersist2 is derived from the variant ICBAPersist and includes such the attributes and services of ICBAPersist and additional attributes and services for Base Object Version.Major greater or equal to 2.

Dispatchable
This attribute contains the availability of type information for the FAL class that is provided by the IDispatch common interface. The Physical Device class provides that kind of information.

Attribute type: Boolean

Allowed values: 1

7.3.2.1.3 Physical device behavior

7.3.2.1.3.1 Overview

For distributed automation FAL ASE object instances, the AR-CoCreateInstance service is only permitted for the PhysicalDevice class. All other objects do not support this service and are "noncreatable".

The PhysicalDevice class shall be implemented according to the Singleton pattern.

NOTE Singleton is a class that possesses exactly one object copy that was instantiated beforehand, and provides a well-defined access point to it. The Singleton pattern ensures that only one instance of a class can be created, and that this instance is globally accessible. The function instance(s) is (are) used to access the unique instance of the class.

The PhysicalDevice class is instantiated by the AR-CoCreateInstance service with the ClassID (UUID_PhysicalDevice), that is well-defined by the distributed automation specification, and an IP address. This service returns an Interface Pointer to the – according to the Singleton pattern – previously instantiated object (i.e. PDev). Any other object of the RT object model can only be reached via navigation procedure (see 7.1.6).

7.3.2.1.3.2 Online/offline comparison of the LDev objects

The PDev stamp reflects changes regarding the associated LDev objects. It is supported through the attribute PDevStamp and will alter, if one of the associated LDev objects is being removed, a new one is being added or an LDev objects name changed.

The PDev stamp is computed as a 32 Bit CRC over all LDev object names, concatenated together with their terminating null character.

NOTE The CRC-32 is also used in Ethernet, PKZip, GZIP, etc. and defined in the following documents: RFC 1952 "GZIP file format specification version 4.3"; ISO Information Processing Systems - Data Communication High-Level; Data Link Control Procedure - Frame Structure", IS 3309, October 1984, 3rd Edition; ITU-T V.42.

The formula for the 32 Bit CRC calculations is defined as follows:

$$r = \mathrm{Crctab32}[(r \,\hat{}\, c) \,\&\, 0\mathrm{xff}] \,\hat{}\, (c >> 8) \qquad \textbf{(20)}$$

WHERE

R	REPRESENTS THE 32 BIT CRC RESULT. THE INITIAL VALUE OF R IS "-1"
C	REPRESENTS THE CURRENT BYTE VALUE THAT NEEDS CRC CALCULATION
CRCTAB32[]	IS DEFINED IN TABLE 127, WHEREAS THE INDEX OF THE APPROPRIATE ENTRY CAN BE FOUND BY ADDING THE NUMBER IN ROW AND COLUMN.

Table 127 – CRC table for the PDev stamp calculation (hexadecimal values)

	0	1	2	3	4	5	6	7
0	00000000	77073096	EE0E612C	990951BA	076DC419	706AF48F	E963A535	9E6495A3
8	0EDB8832	79DCB8A4	E0D5E91E	97D2D988	09B64C2B	7EB17CBD	E7B82D07	90BF1D91
16	1DB71064	6AB020F2	F3B97148	84BE41DE	1ADAD47D	6DDDE4EB	F4D4B551	83D385C7
24	136C9856	646BA8C0	FD62F97A	8A65C9EC	14015C4F	63066CD9	FA0F3D63	8D080DF5
32	3B6E20C8	4C69105E	D56041E4	A2677172	3C03E4D1	4B04D447	D20D85FD	A50AB56B
40	35B5A8FA	42B2986C	DBBBC9D6	ACBCF940	32D86CE3	45DF5C75	DCD60DCF	ABD13D59
48	26D930AC	51DE003A	C8D75180	BFD06116	21B4F4B5	56B3C423	CFBA9599	B8BDA50F
56	2802B89E	5F058808	C60CD9B2	B10BE924	2F6F7C87	58684C11	C1611DAB	B6662D3D
64	76DC4190	01DB7106	98D220BC	EFD5102A	71B18589	06B6B51F	9FBFE4A5	E8B8D433
72	7807C9A2	0F00F934	9609A88E	E10E9818	7F6A0DBB	086D3D2D	91646C97	E6635C01
80	6B6B51F4	1C6C6162	856530D8	F262004E	6C0695ED	1B01A57B	8208F4C1	F50FC457
88	65B0D9C6	12B7E950	8BBEB8EA	FCB9887C	62DD1DDF	15DA2D49	8CD37CF3	FBD44C65
96	4DB26158	3AB551CE	A3BC0074	D4BB30E2	4ADFA541	3DD895D7	A4D1C46D	D3D6F4FB
104	4369E96A	346ED9FC	AD678846	DA60B8D0	44042D73	33031DE5	AA0A4C5F	DD0D7CC9
112	5005713C	270241AA	BE0B1010	C90C2086	5768B525	206F85B3	B966D409	CE61E49F
120	5EDEF90E	29D9C998	B0D09822	C7D7A8B4	59B33D17	2EB40D81	B7BD5C3B	C0BA6CAD
128	EDB88320	9ABFB3B6	03B6E20C	74B1D29A	EAD54739	9DD277AF	04DB2615	73DC1683
136	E3630B12	94643B84	0D6D6A3E	7A6A5AA8	E40ECF0B	9309FF9D	0A00AE27	7D079EB1
144	F00F9344	8708A3D2	1E01F268	6906C2FE	F762575D	806567CB	196C3671	6E6B06E7
152	FED41B76	89D32BE0	10DA7A5A	67DD4ACC	F9B9DF6F	8EBEEFF9	17B7BE43	60B08ED5
160	D6D6A3E8	A1D1937E	38D8C2C4	4FDFF252	D1BB67F1	A6BC5767	3FB506DD	48B2364B
168	D80D2BDA	AF0A1B4C	36034AF6	41047A60	DF60EFC3	A867DF55	316E8EEF	4669BE79
176	CB61B38C	BC66831A	256FD2A0	5268E236	CC0C7795	BB0B4703	220216B9	5505262F
184	C5BA3BBE	B2BD0B28	2BB45A92	5CB36A04	C2D7FFA7	B5D0CF31	2CD99E8B	5BDEAE1D
192	9B64C2B0	EC63F226	756AA39C	026D930A	9C0906A9	EB0E363F	72076785	05005713
200	95BF4A82	E2B87A14	7BB12BAE	0CB61B38	92D28E9B	E5D5BE0D	7CDCEFB7	0BDBDF21
208	86D3D2D4	F1D4E242	68DDB3F8	1FDA836E	81BE16CD	F6B9265B	6FB077E1	18B74777
216	88085AE6	FF0F6A70	66063BCA	11010B5C	8F659EFF	F862AE69	616BFFD3	166CCF45
224	A00AE278	D70DD2EE	4E048354	3903B3C2	A7672661	D06016F7	4969474D	3E6E77DB
232	AED16A4A	D9D65ADC	40DF0B66	37D83BF0	A9BCAE53	DEBB9EC5	47B2CF7F	30B5FFE9
240	BDBDF21C	CABAC28A	53B39330	24B4A3A6	BAD03605	CDD70693	54DE5729	23D967BF
248	B3667A2E	C4614AB8	5D681B02	2A6F2B94	B40BBE37	C30C8EA1	5A05DF1B	2D02EF8D

7.3.2.1.4 Physical device service specification

7.3.2.1.4.1 Interface physical device

7.3.2.1.4.1.1 get_Producer

The get_Producer service reads the name of the device producer. The arguments of the service are shown in Table 128. This service returns the attribute Producer.

Table 128 – get_Producer (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the name of the producer.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.1.2 get_Product

The get_Product service reads the name of the device type in a producer-specific and producer-unambiguous way. The arguments of the service are shown in Table 129. This service returns the attribute Product.

Table 129 – get_Product (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the name of the product.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.1.3 get_SerialNo

The get_SerialNo service reads the serial number of the PDev. The arguments of the service are shown in Table 130. This service returns the attribute SerialNo.

Table 130 – get_SerialNo (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the SerialNo.

Parameter type: VARIANT

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_NOTIMPL, E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.1.4 get_ProductionDate

The get_ProductionDate service reads the production date of the PDev. The arguments of the service are shown in Table 131. This service returns the attribute ProductionDate.

Table 131 – get_ProductionDate (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the ProductionDate.

Parameter type: date

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_NOTIMPL, E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.1.5 Revision

The Revision service reads the revision level of the PDev (i.e. the hardware version) as a major and minor revision. The arguments of the service are shown in Table 132.

Table 132 – Revision (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pMajor			M	M(=)
pMinor			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pMajor
This parameter contains the major part of the revision.

Parameter type: short

pMinor
This parameter contains the minor part of the revision.

Parameter type: short

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.1.6 get_LogicalDevice

The get_LogicalDevice service reads the reference to a LDev object via its name. It is used for navigating in the object model. The arguments of the service are shown in Table 133. This service returns one element of the attribute List of LogicalDevice.

Table 133 – get_LogicalDevice (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Name	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
ppLDev			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Name
This parameter contains the name of the referenced LDev object.

Parameter type: BSTR

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

ppLDev
This parameter contains the address of the ICBALogicalDevice interface of the referenced LDev object. Other LDev interfaces can be obtained using the QueryInterface service.

Parameter type: Interface Pointer

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_UNKNOWNOBJECT, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.1.7 Type

The type service reads the implementation type of the runtime source. The arguments of the service are shown in Table 134.

Table 134 – Type (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pMultiApp			M	M(=)
pPROFInetORPCStack			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pMultiApp
This parameter contains whether the device is able to carry multiple distributed automation enabled applications.

Parameter type: VARIANT_BOOL

pPROFInetORPCStack
This parameter contains whether the device is implemented with a certain ORPC stack.

Parameter type: VARIANT_BOOL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.1.8 PROFInetRevision

The PROFInetRevision service reads the runtime source revision as a major, minor, service pack and build revision. The arguments of the service are shown in Table 135.

Table 135 – PROFInetRevision (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pMajor			M	M(=)
pMinor			M	M(=)
pServicePack			M	M(=)
pBuild			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pMajor
This parameter contains the major part of the revision.

Parameter type: short

pMinor
This parameter contains the minor part of the revision.

Parameter type: short

pServicePack
This parameter contains the service pack part of the revision.

Parameter type: short

pBuild
This parameter contains the build part of the revision.

Parameter type: short

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_INVALIDARG, E_POINTER, RPC_*

7.3.2.1.4.1.9 get_PDevStamp

The get_PDevStamp service reads the PDev stamp (see 7.3.2.1.3.2 for details). The arguments of the service are shown in Table 136. This service returns the attribute PDevStamp.

Table 136 – get_PDevStamp (Physical device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the PDevStamp.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPhysicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.2 Interface Browse

7.3.2.1.4.2.1 get_Count

The get_Count service reads the number of elements in the Browse database. The arguments of the service are shown in Table 137. This service returns the attribute Count.

NOTE Non-dispatch clients should use the get_Count2 service.

Table 137 – get_Count (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the number of Browse items the related object possesses.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.1.4.2.2 BrowseItems

The BrowseItems service reads the list of LDev objects related to a PDev object. The arguments of the service are shown in Table 138.

The server is able to control the maximum amount of returned information via the number of items in the SAFEARRAY. If it obtains less than the number of items indicated via the get_Count service, the client shall continue invoking the service for the subsequent items. To do this, it shall advance the Offset parameter accordingly (by adding the number of returned items, beginning with 0).

The client, in turn, can use the MaxReturn parameter to control the maximum number of returned items. The server shall not return more than this number of items. If the client sets this parameter to 0, the server still returns S_OK, but no items in the SAFEARRAY.

NOTE 1 The client should be able to cope with the fact that the object world has changed between invovations of the get_Count and BrowseItems services. A distributed automation device does not maintain any status information.

NOTE 2 Non-dispatch clients should use the BrowseItems2 service to get extended informations.

Table 138 – BrowseItems (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Offset	M	M(=)		
MaxReturn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pItem			M	M(=)
pDataType			U	U(=)
pAccessRight			U	U(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

Offset
This parameter contains the offset of the first LDev item in the List of LogicalDevice that shall be returned.

Parameter type: long

MaxReturn
This parameter contains the maximum number of LDev items that shall be returned.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pItem
This parameter contains the name or list of names of the available LDevs.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_BSTR

pDataType
This parameter is reserved.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

pAccessRight
This parameter is reserved.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, RPC_*

7.3.2.1.4.2.3 get_Count2

The get_Count2 service reads the number of elements in the Browse database according to the specified selector. The arguments of the service are shown in Table 139.

Table 139 – get_Count2 (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Selector	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

Selector
This parameter contains the items to count. Selector “0” is used for reading the number of LDev objects that are related to the referred PDev object.

Parameter type: long

Allowed values: 0

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the number of Browse items.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, CBA_E_INVALIDENUMVALUE, RPC_*

7.3.2.1.4.2.4 BrowseItems2

The BrowseItems2 service reads the list of available items of a PDev object according to the specified selector. The arguments of the service are shown in Table 140.

The server is able to control the maximum amount of returned information via the number of items in the SAFEARRAY. If it obtains less than the number of items indicated via the get_Count service, the client shall continue invoking the service for the subsequent items. To do this, it shall advance the Offset parameter accordingly (by adding the number of returned items, beginning with 0).

The client, in turn, can use the MaxReturn parameter to control the maximum number of returned items. The server shall not return more than this number of items. If the client sets this parameter to 0, the server still returns S_OK, but no items in the SAFEARRAY.

NOTE The client should be able to cope with the fact that the object world has changed between invovations of the get_Count and BrowseItems services. A distributed automation device does not maintain any status information.

Table 140 – BrowseItems2 (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Selector	M	M(=)		
Offset	M	M(=)		
MaxReturn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pItem			M	M(=)
pInfo1			U	U(=)
pInfo2			U	U(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

Selector
This parameter contains the items to browse. Selector "0" is used for reading a list of LDev objects that are related to the referred PDev object.

Parameter type: long

Allowed values: 0

Offset
This parameter contains the offset of the first LDev item in the List of LogicalDevice that shall be returned.

Parameter type: long

MaxReturn
This parameter contains the maximum number of LDev items that shall be returned.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pItem
This parameter contains the name or list of names of the available items that correspondes to the parameter selector.

For selector "0" it contains the available LDev objects.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_BSTR

pInfo1
This parameter contains information corresponding to the parameter selector.

For selector “0” it is empty.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

pInfo2
This parameter contains information corresponding to the parameter selector.

For selector “0” it is empty.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, CBA_E_INVALIDENUMVALUE, RPC_*

7.3.2.1.4.3 Interface Persist

7.3.2.1.4.3.1 Save

The Save service saves all connection information (inside the consumer) of all ACCO ASEs of the automation device. The connection information shall be stored in non volatile memory and shall be available after a power on.

NOTE 1 The Save service ensures the persistence of all connections that are to be maintained persistent. If the Configuration Data Base is changed via one of the services AddConnections, RemoveConnections, ClearConnections or SetActivationState of an ACCO object, the Save service should subsequently be invoked in order to make this change persistent.

NOTE 2 The PDev is responsible to store the connection information of all ACCO ASEs at the same time. The access to the connection information is a local matter.

The arguments of the service are shown in Table 141.

Table 141 – Save (Persist interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPersist interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPersist interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPersist interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, CBA_E_PERSISTRUNNING, CBA_E_DEFECT, CBA_E_NOTAPPLICABLE, CBA_E_SIZEEXCEEDED, RPC_*

7.3.2.1.4.3.2 Save2

The Save2 service saves all connection information (inside the consumer) of all ACCO ASEs of the automation device. On return each logical device is listed with the partial error resulting in saving this particular LDev object. The connection information shall be stored in non volatile memory and shall be available after a power on.

NOTE 1 The Save service ensures the persistence of all connections that are to be maintained persistent. If the Configuration Data Base is changed via one of the services AddConnections, RemoveConnections, ClearConnections or SetActivationState of an ACCO object, the Save service should subsequently be invoked in order to make this change persistent.

NOTE 2 The PDev is responsible to store the connection information of all ACCO ASEs at the same time. The access to the connection information is a local matter.

The arguments of the service are shown in Table 142.

Table 142 – Save2 (Persist interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pLDevName			M	M(=)
pPartialResult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAPersist2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAPersist2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

pLDevName
This parameter contains the list of LDev names that have been saved.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_BSTR

pPartialResult
This parameter contains the hresult that have been returned by the partial saves.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_ERROR

Allowed values for elements (hresult = S_OK): S_OK

Allowed values for elements (hresult = S_FALSE): S_OK, E_OUTOFMEMORY, CBA_E_DEFECT, CBA_E_NOTAPPLICABLE, CBA_E_SIZEEXCEEDED, RPC_*

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAPersist interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_PERSISTRUNNING, E_OUTOFMEMORY, E_POINTER, E_NOTIMPL, CBA_E_SIZEEXCEEDED

7.3.2.2 Logical Device

7.3.2.2.1 Overview

The LogicalDevice object provides general automation functions such as identification or diagnosis, and is used as navigation anchor for all further objects of the automation device. In addition, it is able to navigate to all contained RT-Auto objects by their names and it offers browsing functionality for the names of the RT-Auto objects. A LDev object represents a software package or a firmware package as an autonomous self-contained unit.

To support an easy online layer for an engineering system, the LDev object is not allowed to change its Name, Producer, Product, SerialNo, ProductionDate or Revision attributes without changing the object instance itself. If the LDev object is to be reconfigured in any of those

attributes, the object instance is to be disconnected by using the CoDisconnectObject service and a new object instance is to be created with the new attribute values. The engineering system therefore detects the change through the error code RPC_E_OBJECTDISCONNECTED by using any service on the old disconnected object instance.

7.3.2.2.2 Logical Device class specification

7.3.2.2.2.1 Template

The Logical Device object is described by the following template:

FAL ASE: **AP ASE**
CLASS: **Logical Device**
CLASS ID: **CBA00010-6C97-11D1-8271-00A02442DF7D**
PARENT CLASS: **Base Object**
ATTRIBUTES and SERVICES:

1	(m)	Key Attribute:	Implicite
2	(m)	Attribute:	Interface Logical Device
2.1	(m)	Attribute:	Name
2.2	(m)	OpsService:	get_Name
2.3	(m)	Attribute:	Producer
2.4	(m)	OpsService:	get_Producer
2.5	(m)	Attribute:	Product
2.6	(m)	OpsService:	get_Product
2.7	(m)	Attribute:	SerialNo
2.8	(m)	OpsService:	get_SerialNo
2.9	(m)	Attribute:	ProductionDate
2.10	(m)	OpsService:	get_ProductionDate
2.11	(m)	OpsService:	Revision
2.12	(m)	Attribute:	ACCO
2.13	(m)	OpsService:	get_ACCO
2.14	(m)	Attribute:	List of RT-Auto
2.14.1	(m)	Attribute:	Name
2.14.2	(m)	Attribute:	Interface Pointer
2.15	(m)	OpsService:	get_RTAuto
2.16	(c)	Constraint:	Base Object Version.Major >= 2
2.16.1	(m)	OpsService:	PROFInetRevision
2.16.2	(m)	OpsService:	ComponentInfo
3	(m)	Attribute:	Interface State
3.1	(m)	Attribute:	State
3.2	(m)	OpsService:	get_State
3.3	(m)	OpsService:	Activate
3.4	(m)	OpsService:	Deactivate
3.5	(m)	OpsService:	Reset
3.6	(m)	OpsService:	AdviseState
3.7	(m)	OpsService:	UnadviseState
4	(o)	Attribute:	Interface Time
4.1	(m)	Attribute:	Time
4.2	(m)	OpsService:	get_Time
4.3	(m)	OpsService:	put_Time
5	(m)	Attribute:	Interface Browse
5.1	(m)	Attribute:	Count
5.2	(m)	OpsService:	get_Count
5.3	(m)	OpsService:	BrowseItems

5.4	(c)	Constraint:	Base Object Version.Major >= 2
5.4.1	(m)	Attribute:	Count2
5.4.2	(m)	OpsService:	get_Count2
5.4.3	(m)	OpsService:	BrowseItems2
6	(m)	Attribute:	Interface Group Error
6.1	(m)	OpsService:	GroupError
6.2	(m)	OpsService:	AdviseGroupError
6.3	(m)	OpsService:	UnadviseGroupError
7	(m)	Attribute:	Dispatchable

7.3.2.2.2.2 Attributes

Interface Logical Device
This attribute contains the logical device interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBALogicalDevice includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBALogicalDevice2 is derived from the variant ICBALogicalDevice and includes such the attributes and services of ICBALogicalDevice and additional attributes and services for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

Name
This attribute contains the name of the LDev.

Attribute type: BSTR

Producer
This attribute contains the name of the device producer.

Attribute type: BSTR (maximum length of 128 usable characters)

Product
This attribute contains the name of the device type in a producer-specific and producer-unambiguous way.

Attribute type: BSTR (maximum length of 128 usable characters)

SerialNo
This attribute contains the serial number of the Logical Device.

Attribute type: VARIANT

ProductionDate
This attribute contains the production date of the Logical Device.

Attribute type: date

ACCO
This attribute contains the local address of the IUnknown interface of the related ACCO object and is used for navigating in the object model.

Attribute type: Interface Pointer

List of RT-Auto
This attribute contains a list of references to all related RT-Auto objects. A single reference can be read via the service get_RTAuto. It is used for navigating in the object model. The attribute is composed of the following list elements.

Name
This attribute contains the name of the RT-Auto object.

Attribute type: BSTR

Interface Pointer
This attribute contains the local address of the ICBARTAuto interface of the RT-Auto object.

Attribute type: Interface Pointer

Interface State
This attribute contains the state interface of the object. It exists in one variant named ICBAState. It is a structured attribute composed of the following attributes.

State
This attribute contains the operating state of the Logical Device.

Attribute type: STATEDEF

The allowed operating states are:

CBANonExistent
The device is without power, communication is impossible to the device. This operating state is only viewable for proxies.

CBADefect
There is a malfunction in PDev or LDev that cannot be eliminated without intervention. Typical defects are hardware or firmware faults. A restart of the LDev is only possible by cycling the power off and back on. Depending on the type of defect, communication with the LDev may be restricted.

CBAInitializing
The LDev performs its initialization after power off/on or reset. The LDev stays in this state until the automation device holds all necessary parameter or configuration data (e.g. self-containing or via download).

CBAReady
The LDev object is ready for operation; the inner activity of the RT-Auto objects is stopped. The outputs of the LDev object are in a safe state.

CBAOperating
The LDev object performs its normal operation. The RT-Auto objects are free with respect to their inner activity.

NOTE 1 The defined operating states may be extended with substates if necessary. The substates may be conveyed by means of manufacturer specific extensions deploying a custom interface.

NOTE 2 The client may wish to be automatically informed about state changes of a LDev. Therefore, it registers itself by means of a AdviceState service delivering an Interface Pointer referencing a client specific StateEvent interface. This interface is beyond the scope of this standard. However, it is assumed that the client provides an "OnStateChanged" service with the "NewState" and "OldState" as service parameters that the server calls in case of a state change.

Interface Time
This attribute contains the time interface of the object. It exists in one variant named ICBATime. It is a structured attribute composed of the following attributes.

Time
This attribute contains the time and date of the Logical Device.

Attribute type: date

Interface Browse
This attribute contains the Browse interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBABrowse includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBABrowse2 is derived from the variant ICBABrowse and includes such the attributes and services of ICBABrowse and additional attributes and services for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

Count
This attribute contains the number of elements in the Browse database.

Attribute type: LONG

Count2
This attribute contains the number of elements in the Browse database.

Attribute type: long

Interface Group Error
This attribute contains the common diagnosis interface of the object. It exists in one variant named ICBAGroupError.

NOTE The client may wish to be automatically informed about group error changes. Therefore, it registers itself by means of the AdviceGroupError service delivering an Interface Pointer referencing a client specific GroupErrorEvent interface. This interface is beyond the scope of this standard. However, it is assumed that the client provides an "OnGroupErrorChanged" service with the "NewState" and "OldState" as service parameters that the server calls in case of a group error change.

Dispatchable
This attribute contains the availability of type information for the FAL Class that is provided by the IDispatch common interface. The Logical Device class provides that kind of information.

Attribute type: Boolean

Allowed values: 1

7.3.2.2.3 Logical Device behavior

7.3.2.2.3.1 Operating state

The operating state graph shown in Figure 17 describes the state behavior of the LDev. The ICBAState interface supports the reading of the current operating state by means of the get_State service. State transitions can be triggered by means of the services Activate, Deactivate, and Reset. The meaning of the particular states is specified in 7.3.2.2.2.2.

The operating state of the LDev affects the behavior of the RT-Auto objects. The RT-Auto objects shall only be active ("running") in the LDev state "CBAOperating". In all other states the activity of the RT-Auto objects is stopped. Therefore, the outputs of the LDev related RT-Auto objects shall be in the safe state. This definition does not apply to System-RT-Auto objects. System-RT-Auto objects are independent of the operating state.

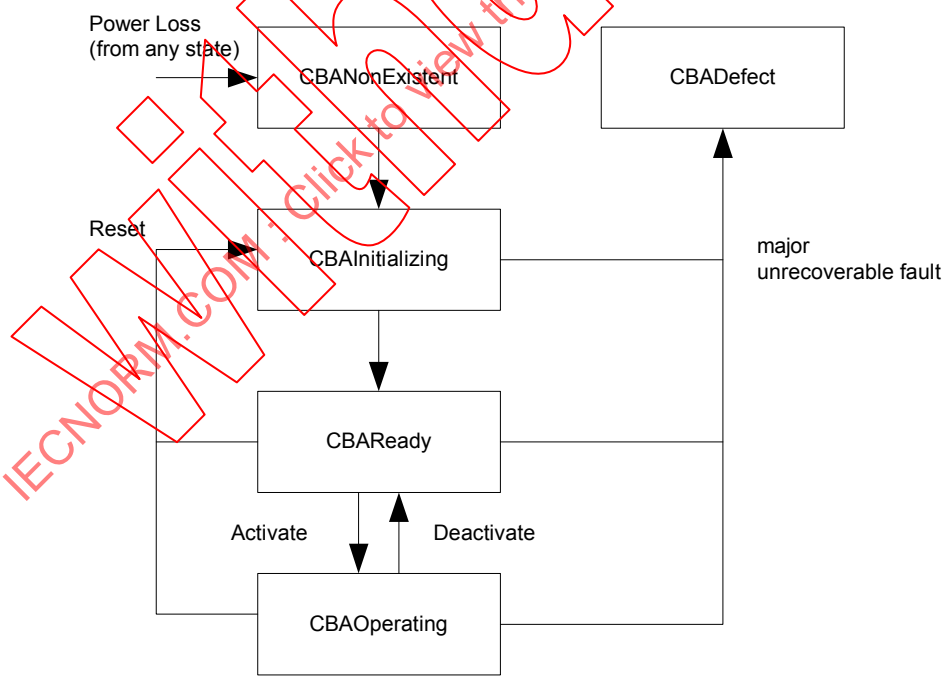


Figure 17 – Operating state block diagram

NOTE The defined operating states may be extended with substates if necessary. The substates may be conveyed by means of manufacturer specific extensions deploying a custom interface.

7.3.2.2.3.2 Diagnosis

The common diagnosis contains the diagnoses contents and the access of the diagnosis information via services at the ICBAGroupError interface. Figure 18 shows the common diagnoses model.

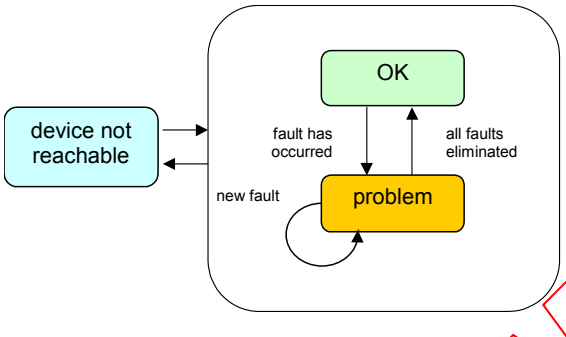


Figure 18 – Device status model for the common diagnosis

The common diagnosis is the top level diagnosis information of the device. On this level, there is no further categorization in corresponding error types.

NOTE 1 The common diagnosis of a device corresponds for example to a common fault LED of a controller.

The common diagnosis can be fetched via the GroupError service. This service reads the current error status and a change indicator referred to as cookie. The cookie can be used by the client to determine the changing of the group error. The cookie itself shall be modified by the server object (e.g. increment) if the group error changes.

NOTE 2 An external client should not derive any information from the internal cookie generation procedure. The cookie is not hold persistent over the power off of the unit, i.e. the same cookie can appear before power off and after power off by different error states. Since the ICBAGroupError interface instance is invalid after power on, a client should easily be able to detect this condition.

7.3.2.2.4 Logical Device service specification

7.3.2.2.4.1 Interface logical device

7.3.2.2.4.1.1 get_Name

The get_Name service reads the name of the Logical Device. The name is used for navigation. The arguments of the service are shown in Table 143. This service returns the attribute Name.

Table 143 – get_Name (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the name of the Logical Device.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.1.2 get_Producer

The get_Producer service reads the name of the device producer. The arguments of the service are shown in Table 144. This service returns the attribute Producer.

Table 144 – get_Producer (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the name of the producer.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.1.3 get_Product

The get_Product service reads the name of the device type in a producer-specific and producer-unambiguous way. The arguments of the service are shown in Table 145. This service returns the attribute Product.

Table 145 – get_Product (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the name of the product.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.1.4 get_SerialNo

The get_SerialNo service reads the serial number of the LDev. The arguments of the service are shown in Table 146. This service returns the attribute SerialNo.

Table 146 – get_SerialNo (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the SerialNo.

Parameter type: VARIANT

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_NOTIMPL, E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.1.5 get_ProductionDate

The get_ProductionDate service reads the production date of the LDev. The arguments of the service are shown in Table 147. This service returns the attribute ProductionDate.

Table 147 – get_ProductionDate (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the ProductionDate.

Parameter type: date

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_NOTIMPL, E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.1.6 Revision

The Revision service reads the revision level of the LDev (i.e. the software version) as a major and minor revision. The arguments of the service are shown in Table 148.

NOTE For a type 3 slave the method always returns “0.0”, whereas this is seen as an invalid revision number. Therefore a client is able to use the method as a simple detection whether the LDev is still available in the sense of communication.

Table 148 – Revision (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pMajor			M	M(=)
pMinor			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pMajor
This parameter contains the major part of the revision.

Parameter type: short

pMinor
This parameter contains the minor part of the revision.

Parameter type: short

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_POINTER, E_NOTIMPL, RPC_*

7.3.2.2.4.1.7 get_ACCO

The get_ACCO service supports the navigation procedure and reads the attribute ACCO containing a reference to the ACCO object. This reference is deployed to address ACCO services. The arguments of the service are shown in Table 149.

Table 149 – get_ACCO (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
ppACCO			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

ppACCO
This parameter contains the address of the IUnknown interface of the referenced ACCO object. Other interfaces of the ACCO object can be obtained using the QueryInterface service.

Parameter type: Interface Pointer

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.1.8 get_RTAuto

The get_RTAuto service reads the reference to an RT-Auto object via its name and is used for navigating in the object model. The arguments of the service are shown in Table 150. This service returns one element of the attribute List of RT-Auto.

Table 150 – get_RTAuto (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Name	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
ppAuto			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

Name
This parameter contains the name of the referenced RT-Auto object.

Parameter type: BSTR

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

ppAuto
This parameter contains the address of the RT-Auto interface of the referenced RT-Auto object. Other interfaces of the RT-Auto object can be obtained using the QueryInterface service.

Parameter type: Interface Pointer

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, CBA_E_UNKNOWNOBJECT, E_POINTER, RPC_*

7.3.2.2.4.1.9 PROFInetRevision

The PROFInetRevision service reads the runtime source revision as a major, minor, service pack and build revision. The arguments of the service are shown in Table 151.

NOTE Only for a multi-application device this service may return different information than the PROFInetRevision service of the Physical device interface.

Table 151 – PROFInetRevision (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pMajor			M	M(=)
pMinor			M	M(=)
pServicePack			M	M(=)
pBuild			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pMajor
This parameter contains the major part of the revision.

Parameter type: short

pMinor
This parameter contains the minor part of the revision.

Parameter type: short

pServicePack
This parameter contains the service pack part of the revision.

Parameter type: short

pBuild
This parameter contains the build part of the revision.

Parameter type: short

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_INVALIDARG, E_POINTER, RPC_*

7.3.2.2.4.1.10 ComponentInfo

The ComponentInfo service reads the component information of the LDev. The arguments of the service are shown in Table 152.

NOTE The ComponentID reflects the engineering component's ID – the one of the component's description.

Table 152 – ComponentInfo (Logical Device interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pComponentID			M	M(=)
pVersion			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pComponentID
This parameter contains the Component identifier.

Parameter type: BSTR

pVersion
This parameter contains the version.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBALogicalDevice2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.2 Interface State

7.3.2.2.4.2.1 get_State

The get_State service reads the current operating state of a LDev object. The arguments of the service are shown in Table 153.

NOTE The operating states "CBANonExistent" and "CBADefect" can only be monitored partially. An LDev object that is currently not ready to communicate is typically advised via the result "RPC_S_SERVER_UNAVAILABLE". This permits the operating state to be derived implicitly.

This service returns the attribute State.

Table 153 – get_State (State interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the state of the Logical Device.

Parameter type: STATEDEF

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.2.2 Activate

The Activate service is used to set a LDev to the "CBAOperating" state. This is only possible if the LDev is in "CBAReady" state and there is no specific startup obstacle (e.g. the keyswitch of a PLC is on STOP position).

NOTE The service checks the startup obstacles, but only initiates the state change during this process. The state change proper is performed asynchronously with the service invocation. A client that wants to determine the success should log on for state transitions. Once the state transition has been completed, it will be reported asynchronously there.

The arguments of the service are shown in Table 154.

Table 154 – Activate (State interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request..

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_NOTAPPLICABLE, E_INVALIDARG, E_OUTOFMEMORY, RPC_*

7.3.2.2.4.2.3 Deactivate

The Deactivate service is used to set the LDev to the "CBAReady" state. This is only possible if the LDev is in "CBAOperating" state.

NOTE The service merely initiates the state change. The state change proper is performed asynchronously with the service invocation. A client that wants to determine the success should log on for state transitions. Once the state transition has been completed, it will be reported asynchronously there.

The arguments of the service are shown in Table 155.

Table 155 – Deactivate (State interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request..

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_NOTAPPLICABLE, E_INVALIDARG, E_OUTOFMEMORY, RPC_*

7.3.2.2.4.2.4 Reset

The Reset service is used to reset the LDev to the "CBAInitializing" state. This is only possible if the LDev is not in "CBADefect" state.

NOTE The service merely initiates the state change. The state change proper is performed asynchronously with the service invocation. A client that wants to determine the success should log on for state transitions. Once the state transition has been completed, it will be reported asynchronously there.

The arguments of the service are shown in Table 156.

Table 156 – Reset (State interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request..

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_NOTAPPLICABLE, E_INVALIDARG, E_OUTOFMEMORY, RPC_*

7.3.2.2.4.2.5 AdviseState

The AdviseState service requests the server to inform the client if the attribute State of a LDev object changes.

NOTE 1 The client may wish to be automatically informed about state changes of a LDev. Therefore, it registers itself by means of a AdviceState service delivering an Interface Pointer referencing a client specific StateEvent interface. This interface is beyond the scope of this standard. However, it is assumed that the client provides an "OnStateChanged" service with the "NewState" and "OldState" as service parameters that the server calls in case of a state change.

The arguments of the service are shown in Table 157.

Table 157 – AdviseState (State interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
pStateEvent	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pCookie			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

pStateEvent
This parameter contains the address of the event interface.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

NOTE 2 The value S_FALSE is returned if the transferred interface pointer has already been registered as an event interface. The cookie contains the interface pointer of the previous registration. This means, that the interface pointer is not registered again in this case.

pCookie
This parameter contains the event interface handle.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.2.6 UnadviseState

The UnadviseState service clears the event advise. The arguments of the service are shown in Table 158.

Table 158 – UnadviseState (State interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Cookie	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

Cookie
This parameter contains the handle to the event interface.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAState interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, CBA_E_INVALIDCOOKIE, RPC_*

7.3.2.2.4.3 Interface Time

7.3.2.2.4.3.1 get_Time

The get_Time service reads the current time of a LDev. The arguments of the service are shown in Table 159. This service returns the attribute Time.

Table 159 – get_Time (Time interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBATime interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBATime interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the current time of the LDev object.

Parameter type: date

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBATime interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_NOTIMPL, E_POINTER, RPC_*

7.3.2.2.4.3.2 put_Time

The put_Time service sets the current time of a LDev. The arguments of the service are shown in Table 160. This service writes the attribute Time.

Table 160 – put_Time (Time interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
NewVal	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBATime interface of the requested object instance.

Parameter type: Interface Pointer

NewVal
This parameter contains the new time for the LDev object.

Parameter type: date

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBATime interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBATime interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_LIMITVIOLATION, CBA_E_TIMEVALUEUNSUPPORTED, E_ACCESSDENIED, E_INVALIDARG, E_OUTOFMEMORY, E_NOTIMPL, RPC_*

7.3.2.2.4.4 Interface Browse

7.3.2.2.4.4.1 get_Count

The get_Count service reads the number of elements in the Browse database. The arguments of the service are shown in Table 161. This service returns the attribute Count.

NOTE Non-dispatch clients should use the get_Count2 service.

Table 161 – get_Count (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M (=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the number of elements.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.4.2 BrowseItems

The BrowseItems service reads the list of RT-Auto objects related to a LDev object. The arguments of the service are shown in Table 162.

The server is able to control the maximum amount of returned information via the number of items in the SAFEARRAY. If it obtains less than the number of items indicated via the get_Count service, the client shall continue invoking the service for the subsequent items. To do this, it shall advance the Offset parameter accordingly (by adding the number of returned items, beginning with 0).

The client, in turn, can use the MaxReturn parameter to control the maximum number of returned items. The server shall not return more than this number of items. If the client sets this parameter to 0, the server still returns S_OK, but no items in the SAFEARRAY.

NOTE 1 The client should be able to cope with the fact that the object world has changed between invovations of the get_Count and BrowseItems services. A distributed automation device does not maintain any status information.

NOTE 2 Non-dispatch clients should use the BrowseItems2 service to get extended informations.

Table 162 – BrowseItems (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Offset	M	M(=)		
MaxReturn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pItem			M	M(=)
pDataType			U	U(=)
pAccessRight			U	U(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

Offset
This parameter contains the offset of the first item in the List of RT-Auto that shall be returned.

Parameter type: long

MaxReturn
This parameter contains the maximum number of RT-Auto items that shall be returned.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pItem
This parameter contains the name or list of names of the available RT-Auto objects.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_BSTR

pDataType
This parameter is reserved.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

pAccessRight
This parameter is reserved.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, RPC_*

7.3.2.2.4.4.3 get_Count2

The get_Count2 service reads the number of elements in the Browse database according to the specified selector. The arguments of the service are shown in Table 163. This service reads the attribute Count.

Table 163 – get_Count2 (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Selector	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

Selector
This parameter contains the items to count. Selector “0” is used for reading the number of RT-Auto objects and selector "1" for the number of System RT-Auto objects that are related to the referred LDev object.

Parameter type: long

Allowed values: 0, 1

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the number of Browse items.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, CBA_E_INVALIDENUMVALUE, RPC_*

7.3.2.2.4.4.4 BrowseItems2

The BrowseItems2 service reads the list of available items of a LDev object according to the specified selector. The arguments of the service are shown in Table 164.

The server is able to control the maximum amount of returned information via the number of items in the SAFEARRAY. If it obtains less than the number of items indicated via the get_Count service, the client shall continue invoking the service for the subsequent items. To do this, it shall advance the Offset parameter accordingly (by adding the number of returned items, beginning with 0).

The client, in turn, can use the MaxReturn parameter to control the maximum number of returned items. The server shall not return more than this number of items. If the client sets this parameter to 0, the server still returns S_OK, but no items in the SAFEARRAY.

NOTE The client should be able to cope with the fact that the object world has changed between invovations of the get_Count and BrowseItems services. A distributed automation device does not maintain any status information.

Table 164 – BrowseItems2 (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Selector	M	M(=)		
Offset	M	M(=)		
MaxReturn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pItem			M	M(=)
pInfo1			U	U(=)
pInfo2			U	U(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

Selector
This parameter contains the items to browse. Selector “0” is used for reading a list of RT-Auto objects and selector "1" for a list of System RT-Auto objects that are related to the referred LDev object.

Parameter type: long

Allowed values: 0, 1

Offset
This parameter contains the offset of the first LDev item in the List of LogicalDevice that shall be returned.

Parameter type: long

MaxReturn
This parameter contains the maximum number of LDev items that shall be returned.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pItem
This parameter contains the name or list of names of the available items that correspondes to the parameter selector.

For selector “0” it contains the RT-Auto objects of the LDev object.

For selector “1” it contains the system RT-Auto object.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_BSTR

pInfo1
This parameter contains information corresponding to the parameter selector.

For selector “0” and "1" it is empty.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

pInfo2
This parameter contains information corresponding to the parameter selector.

For selector “0” and "1" it is empty.

Parameter type: VARIANT

Allowed values: Tag=VT_ERROR, Value=DISP_E_PARAMNOTFOUND.

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, CBA_E_INVALIDENUMVALUE, RPC_*

7.3.2.2.4.5 Interface group error

7.3.2.2.4.5.1 GroupError

The GroupError service reads the error state of a LDev object. The returned cookie contains whether or not the group error state of the device has changed since the last invocation. The arguments of the service are shown in Table 165.

Table 165 – GroupError (Group error interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
pMagicCookie			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the group error state of the LDev object.

Parameter type: GROUPERRORDEF

pMagicCookie
This parameter contains the group error state change indicator.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.5.2 AdviseGroupError

The AdvisegroupError service requests the server to inform the client if the group error changes.

NOTE 1 This service is depreciated for devices with Base Object Version 2 or later. Use a connection to the state collection at the System RT-Auto object instead.

NOTE 2 The client may wish to be automatically informed about group error changes of a LDev object. Therefore, it registers itself by means of a AdviceGroupError service delivering an Interface Pointer referencing a client specific GroupErrorEvent interface. This interface is beyond the scope of this standard. However, it is assumed that the client provides an "OnGroupErrorChanged" service with the "NewState" and "OldState" as service parameters that the server calls in case of a group error change.

The arguments of the service are shown in Table 166.

Table 166 – AdviseGroupError (Group Error interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
pGroupErrorEvent	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pCookie			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

pGroupErrorEvent
This parameter contains the address of the event interface.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

NOTE 3 The value S_FALSE is returned if the transferred interface pointer has already been registered as an event interface. The cookie contains the interface pointer of the previous registration. This means, that the interface pointer is not registered again in this case.

pCookie
This parameter contains the event interface handle.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.2.2.4.5.3 UnadviseGroupError

The UnadviseGroupError service clears the event advise referenced by cookie. The arguments of the service are shown in Table 167.

NOTE This service is depreciated for devices with Base Object Version 2 or later. Use a connection to the state collection at the System RT-Auto object instead.

Table 167 – UnadviseGroupError (Group Error interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Cookie	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

Cookie
This parameter contains the handle to the event interface.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, CBA_E_INVALIDCOOKIE, RPC_*

7.3.3 ACCO ASE

7.3.3.1 Overview

The ACCO (Active Control Connection Object) objects implements a configurable interconnection of RT-Auto objects.

7.3.3.2 ACCO class specification

7.3.3.2.1 Template

The ACCO object is described by the following template:

FAL ASE:	**ACCO ASE**		
CLASS:	**ACCO**		
CLASS ID:	**CBA00040-6C97-11D1-8271-00A02442DF7D**		
PARENT CLASS:	**Base Object**		
ATTRIBUTES and SERVICES:			
1	(m)	Key Attribute:	Implicite
2	(m)	Attribute:	Interface ACCO Management
2.1	(m)	OpsService:	AddConnections
2.2	(m)	OpsService:	RemoveConnections
2.3	(m)	OpsService:	ClearConnections
2.4	(m)	OpsService:	SetActivationState
2.5	(m)	OpsService:	GetInfo
2.6	(m)	OpsService:	GetIDs
2.7	(m)	OpsService:	GetConnections
2.8	(m)	OpsService:	ReviseQoS
2.9	(m)	Attribute:	PingFactor
2.10	(m)	OpsService:	get_PingFactor
2.11	(m)	OpsService:	put_PingFactor
2.12	(m)	Attribute:	CDBCookie
2.13	(m)	OpsService:	get_CDBCookie
2.14	(c)	Constraint:	Base Object Version.Major >= 2
2.14.1	(m)	OpsService:	GetConsIDs
2.14.2	(m)	OpsService:	GetConsConnections
2.14.3	(m)	OpsService:	DiagConsConnections
2.14.4	(m)	OpsService:	GetProvIDs
2.14.5	(m)	OpsService:	GetProvConnections
2.14.6	(m)	OpsService:	GetDiagnosis
3	(m)	Attribute:	Interface ACCO Server
3.1	(m)	OpsService:	Connect
3.2	(m)	OpsService:	Disconnect
3.3	(m)	OpsService:	DisconnectMe
3.4	(m)	OpsService:	SetActivation
3.5	(m)	OpsService:	Ping
3.6	(c)	Constraint:	Base Object Version.Major >= 2
3.6.1	(m)	OpsService:	Connect2
3.6.2	(m)	OpsService:	GetConnectionData
4	(m)	Attribute:	Interface ACCO Callback
4.1	(m)	OpsService:	OnDataChanged
4.2	(c)	Constraint:	Base Object Version.Major >= 2
4.2.1	(m)	OpsService:	Gnip
5	(m)	Attribute:	Interface ACCO Sync
5.1	(m)	OpsService:	ReadItems

5.2	(m)	OpsService:	WriteItems
5.3	(m)	OpsService:	WriteItemsQCD
6	(m)	Attribute:	Interface Group Error
6.1	(m)	OpsService:	GroupError
6.2	(m)	OpsService:	AdviseGroupError
6.3	(m)	OpsService:	UnadviseGroupError
7	(c)	Constraint:	Base Object Version.Major >= 2
7.1	(o)	Attribute:	Interface ACCO Server SRT
7.1.1	(m)	OpsService:	ConnectCR
7.1.2	(m)	OpsService:	DisconnectCR
7.1.3	(m)	OpsService:	Connect
7.1.4	(m)	OpsService:	Disconnect
7.1.5	(m)	OpsService:	DisconnectMe
7.1.6	(m)	OpsService:	SetActivation
8	(m)	Attribute:	Dispatchable

7.3.3.2.2 Attributes

Interface ACCO Management

This attribute contains the configuration data base interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBAAccoMgt includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBAAccoMgt2 is derived from the variant ICBAAccoMgt and includes such the attributes and services of ICBAAccoMgt and additional attributes and services for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

PingFactor

This attribute contains the current PingFactor that is used for determining the maximum dead time of a provider (PingFactor * minimum QoS).

Attribute type: unsigned short

Allowed values are shown in Table 168.

Table 168 – PingFactor values

PingFactor	Description
0 – 65 535	valid values
10	default value
0	disable pinging

CDBCookie

This attribute contains the Configuration Data Base change indicator.

Attribute type: long

Interface ACCO Server

This attribute contains the connection provider interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBAAccoServer includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBAAccoServer2 is derived from the variant ICBAAccoServer and includes such the attributes and services of ICBAAccoServer and additional attributes and services for Base Object Version.Major greater or equal to 2.

Interface ACCO Callback

This attribute contains the connection consumer interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBAAccoCallback includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBAAccoCallback2 is derived from the variant ICBAAccoCallback and includes such the attributes and services of ICBAAccoCallback and additional attributes and services for Base Object Version.Major greater or equal to 2.

Interface ACCO Sync
This attribute contains the synchronous reading or writing interface of the object. It exists in one variant named ICBAAccoSync.

Interface Group Error
This attribute contains the common diagnosis interface of the object. It exists in one variant named ICBAGroupError.

NOTE The client may wish to be automatically informed about group error changes. Therefore, it registers itself by means of the AdviceGroupError service delivering an Interface Pointer referencing a client specific GroupErrorEvent interface. This interface is beyond the scope of this standard. However, it is assumed that the client provides an "OnGroupErrorChanged" service with the "NewState" and "OldState" as service parameters that the server calls in case of a group error change.

Interface ACCO Server SRT
This attribute contains the RT connection provider interface of the object. It exists in one variant named ICBAAccoServerSRT.

Dispatchable
This attribute contains the availability of type information for the FAL Class that is provided by the IDispatch common interface. The ACCO ASE does not provide that kind of information and the Dispatchable attribute contains the value 0.

Attribute type: Boolean

7.3.3.3 ACCO Behavior

7.3.3.3.1 Interconnecting RT-Auto objects – ACCO

The ACCO (Active Control Connection Object) is a general purpose FAL Class that shall have only one object instance per LDev. The main purpose of this class is to provide optimized and dynamically configurable (during runtime) data exchange between automation task specific attributes (inside custom interfaces) of the RT-Auto objects. These class attributes are not part of this specification because they are automation task specific. The ACCO ASE defines a way to collect such attribute values from the local RT-Auto objects (local matter or local object behavior) and provide the attribute values to remote RT-Auto objects via the remote ACCO object. Therefore, the ACCO ASE introduces a consumer and provider facet. The ACCO ASE offers connection management services to define such connections during runtime. It conveys data between RT-Auto objects using connection information. The access to the RT-Auto object data items is local matter.

An RT-Auto ASE offers data (attributes or properties) and events (services or methods) as information. Consequently, the ACCO ASE offers data connections.

7.3.3.3.2 Architecture

The ACCO ASE is responsible for the following tasks:

– Establishing configured or dynamically created (for OS connections, for example) connections via symbolic addressing with the properties
– Transportation of data and events along these connections with the configured quality (Quality of Service)
– Ensuring the transparency of the connection between RT-Auto objects (local and remote are handled identically)
– Implementing failure strategies for data (specific to the individual runtime environments)
– Optimizing the transport of data
– Enabling various communication procedures (and their expansions) to be used - also under the aspect of real-time communication procedures
– Implementing simple operators to data connections

The following terms are used in the description:

– A Provider is the source of the data. It produces the information of a connection.
– A Consumer is the sink of the data. It receives the information of a connection.

The basic rule of a connection defines that the consumer shall always ensure that the information is obtained; the provider merely makes the information available. This has an effect on the configuration of the connections (always at the consumer's device) and on the fault strategy.

Figure 19 shows the internal structure of the ACCO ASE and the operation in relation to object instances of the RT-Auto ASE.

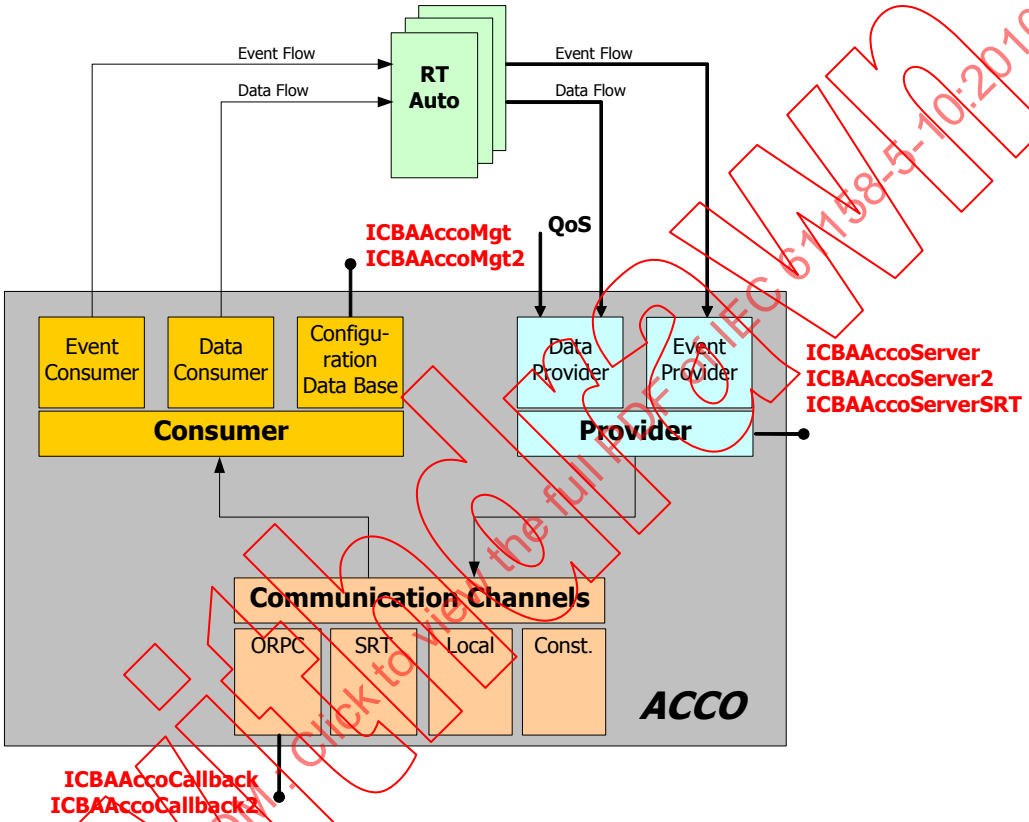


Figure 19 – ACCO ASE structure

The ACCO ASE consists of the following components:

– The Consumer provides the functions required for configuring connections and connection sinks.
– The Provider provides the functions required for connection sources.
– The Communication abstracts the deployed communication path for provider and consumer

7.3.3.3.3 Short overview

7.3.3.3.3.1 Establishing connections

A configuration tool transfers the connection information to the ACCO that carries the consumer of some connections to establish (connection sinks). The consumer then negotiates with the provider the productive operation of the connections involved.

7.3.3.3.3.2 Productive operation of data connections

During the productive operation, the provider transmits the connection data via the negotiated com-munication channel to the consumer.

For data connections, this works according to the block diagram in Figure 20.

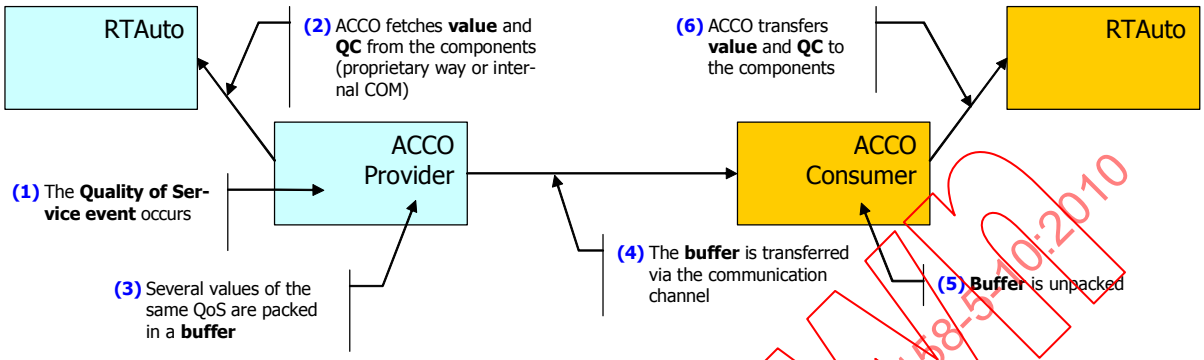


Figure 20 – Productive operation of data connections

For data connections, productive operation is performed in the following steps:

1. An event for the configured Quality of Service (e.g. a time interval of 100 ms expires) occurs at the provider. The way in which that is signaled to the provider depends on the deployed QoS.
2. The connection data is read via the access methods at the RT-Auto objects. The time to read the connection data is dependent on the communication channel used:
   – For ORPC channels the connection data are read when the QoS event triggers the provider.
   – For local and RT channels the read of connection data is triggered by the application.
1. The connection data are filtered for changes (applies only to the ORPC channel) and packed into a buffer.
2. The data item is transferred to the consumer via the communication channel.
3. The buffer is unpacked.
4. The connection data are transferred to the RT-Auto objects. The write of connection data is done differently according to the communication channel used:
   – For ORPC channels the connection data are written when the buffer is unpacked.
   – For local and RT channels the connection data is written at an application specific time.

7.3.3.3.4 Definitions

7.3.3.3.4.1 Configuration Data Base

The Configuration Data Base holds the information about the connections between the individual RTAuto objects that is generated by an engineering system. It is transferred via the ACCO Management interface. Only the ACCO object of a configuration sink obtains the configuration data.

A data connection between two RT-Auto objects is characterized by the following information:

– Identification of the connection source (i.e. the identification of the property or event at the RT-Auto object of a specific LDev – see 7.3.3.3.4.2),

– Identification of the connection sink (see also 7.3.3.3.4.2),
– Initial state of the connection (active or inactive); i.e. does the connection run immediately or only after an explicit activation (see 7.3.3.3.6.5).

A data connection requires the following additional information:

– Quality of Service is characterized by type and qualifier; i.e. when data are to be transferred (see 7.3.3.3.4.4),
– Epsilon value for delimiting the transmission frequency, e.g. for analog values (see 7.3.3.3.4.5),
– Substitute value, for applying specific values in case of a failure of the communication relationship (see 7.3.3.3.4.7).

If the initial state of the connection is inactive, the connection is actually established but productive operation will not take place before the connection has been activated by the SetActivationState service provided by the ACCO Management interface.

7.3.3.3.4.2 Rules for connections

A connection defines a link between a data or event source and the corresponding data or method sink and shall obey the following rules.

– A connection can only be established between attributes (properties), services (events and methods) at custom interfaces of RT-Auto objects. Additionally data connections can be established with a constant as source.
– A connection can only be established between partners of the same signature:
  – Data connections shall use identical data types in source and sink (e.g. signed long).
– A data connection can be established by using the AddConnections service of the ACCO Management interface between two properties of RT-Auto objects, whereas the source of the connection needs a property access service with the IDL attribute [propget]. The sink of the connection needs a property access service with the IDL attribute [propput].

  A data connection can further be established by using the AddConnections service of the ACCO Management interface between a constant and a property of an RT-Auto object, whereas the source of the connection is a constant. The sink of the connection needs a property access service with the IDL attribute [propput].

  A data connection can further be established by using the Connect2 service of the ACCO Sserver interface between a property of an RT-Auto object and for example a HMI application.

  A data connection with a constant as consumer is not supported.
– Merging data (i.e. connecting several data sources with one data sink) is not permitted. Routing multiple events to one service is permitted.
– Splitting a source to several sinks is always possible (for data and for events).
– Interconnecting a source with itself being the sink is not permitted.

7.3.3.3.4.3 Definition of the identifiers

7.3.3.3.4.3.1 Identifier for an LDev object

The identifier of an LDev object is defined by the following string:

"PDev!LDev"

An exclamation mark is used as delimiter.

PDev
is used for identifying the hardware in the IP sense. The IP address can be used here.

LDev
corresponds to the name of the LogicalDevice.

In order to facilitate identification systems of all variants with respect to the nomenclature of the LDev, all characters are permitted for LDev (as mentioned above). This requires a special interpretation of the identifier by first extracting from the left-hand side to the "!" PDev. The remainder is the identification of the LDev.

For a connection with a constant the identifier “CONSTANT!!” is used (see 7.3.3.3.6.2.2 for details).

For the character sets defined for PDev and LDev objects see 7.2.4.4.

7.3.3.3.4.3.2 Identifier for an RT-Auto menber

The identification for the source or sink of a data connection is performed in plain text via an identifier. The identifier unambiguously describes an attribute (property) or service (event or method) in an interface of an RT-Auto Class. The syntax of the identifier has been defined as follows:

"Object.Member"

The delimiter between object and member is a point.

Object
corresponds to the name of the RT-Auto object that can also be used at the enumeration of RT-Auto objects.

In order to facilitate identification systems of all variants with respect to the nomenclature of RT-Auto objects, all characters are permitted for Object. This requires a special interpretation of the identifier in which the method name (defined in the IDL) is extracted from the right-hand side up to the ".". The method name has been defined with a restricted character set. The remainder is the identification of the RT-Auto object.

Member
For a data connection, member unambiguously describes the put or get method used for accessing the corresponding property.

The member follows the identification system that is defined by the IDL.

With respect to possible language variants, only the member name defined under the "Neutral" language (LANG_NEUTRAL) of the Dispatch interface is supported (which is the name supported in the IDL). The configuration of the connections shall therefore be performed in a language-independent way.

An example for an interface declaration for this purpose follows below:

```
EXAMPLE    [
              object,
              uuid(7f41e860-ba54-11d1-912c-00a02442df7d),
              oleautomation,
              dual,
              pointer_default(unique)
           ]
           interface IReactor : IDispatch
           {
              [propput] HRESULT InflowValve([in] VARIANT_BOOL newVal);
              [propget] HRESULT InflowValve([out, retval] VARIANT_BOOL
           *pVal);
              [propput] HRESULT OutflowValve([in] VARIANT_BOOL newVal);
              [propget] HRESULT OutflowValve([out, retval] VARIANT_BOOL
           *pVal);
              [propget] HRESULT Level([out, retval] long *pVal);
           };
```

```
[
   object,
   uuid(7f41e861-ba54-11d1-912c-00a02442df7d),
   oleautomation,
   dual,
   pointer_default(unique)
]
interface ITemp : IDispatch
{
   [propget] HRESULT CurrentTemp([out, retval] float *pVal);
};

...

[
   uuid(7f41e862-ba54-11d1-912c-00a02442df7d),
   noncreatable
]
coclass Reactor
{
   [default]  interface IReactor;
              interface ITemp;
};
```

If there is an RT-Auto object "Reactor 1" of the Reactor class, the source of the "OutflowValve" property (i.e. the propget method) is identified by the designator

```
"Reactor 1.OutflowValve"
```

To interconnect the "CurrentTemp" property, the designator

```
"Reactor 1.CurrentTemp"
```

should be used.

For the character sets defined for RT-Auto objects see 7.2.4.4.

7.3.3.3.4.3.3 Identifier for a System Member

The syntax of the identifier is defined by the following string:

"!SYSTEM.Member"

The delimiter between the system object and member is a point.

!SYSTEM
addresses the System RT-Auto object "SYSTEM".

Member
is defined as above, indicating a member of the System Properties interface

To interconnect the “STATECOLLECTION” property on the ACCO, the following designator is used:

„!SYSTEM.STATECOLLECTION”

To interconnect the “STAMPCOLLECTION” property on the LDev, the designator

„!SYSTEM.STAMPCOLLECTION”

shall be used.

7.3.3.3.4.3.4 Identifiers for accessing sub elements

The identifier is extended in the following way to allow access to sub elements (members of a structure or an array). The notation may only be used by forcing an element and to read or write an element through the services ReadItems and WriteItems of the ACCO Sync interface.

The AccessList for accessing sub elements follows the C notation, whereas specified through the following grammar, which is in its syntax closely related to the C syntax for accessing sub elements of a variable:

AccessList:
ArrayElement

ArrayElement StructureList

, StructureElement

, StructureElement AccessList

StructureList:
, StructureElement

, StructureElement AccessList

StructureElement:
Number

ArrayElement:
[NumberList]

NumberList:
Number

Number, NumberList

Number:
[0-9]+

The AccessList can only be used for items of data type SAFEARRAY or structure.

For an ArrayElement a number describes the zero-based index of the element. It is not possible to access a subarray of the SAFEARRAY with this notation, only base elements of scalar types or strings (VT_BOOL, VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8, VT_DATE and VT_BSTR) are supported. If the array holds structures as elements, the notation follows the definition below.

For a StructureElement the number describes the zero-based position of the element counted by its position in the structure. It is not possible to access a substructure with this notation, only base elements of scalar types or strings are supported. If the structure holds an array as element, the notation for accessing an array element follows the definition above.

Object and Member are used as defined in 7.3.3.3.4.3.2.

Examples for accessing sub elements follow below:

EXAMPLE 1
```
[
    object,
    uuid(7f41e860-ba54-11d1-912c-00a02442df7d),
    oleautomation,
    dual,
    pointer_default(unique)
]
interface IXRay : IDispatch
{
    ...
    [propget] HRESULT XRayData([out, retval] SAFEARRAY *pVal);
};
```

If the reactor above holds a three dimensional SAFEARRAY with base type double holding XRay data, the identifier used to access a specific element would be

```
"Reactor 1.XRayData[17,567,8]"
```

This gives access to the array element XRayData[17,567,8]. Using only two numbers to denote the ArrayElement is not allowed, since this would

address a column in the matrix.

EXAMPLE 2
```
typedef struct {
    double radiation;
    double leakage;
} xraydata;
[
    object,
    uuid(7f41e860-ba54-11d1-912c-00a02442df7d),
    oleautomation,
    dual,
    pointer_default(unique)
]
interface IXRay : IDispatch
{
    ...
    [propget] HRESULT XRayData([out, retval] SAFEARRAY *pVal);
};
```

If the reactor above holds a three dimensional SAFEARRAY but this time with base type xraydata, the identifier used to access a specific element would be

```
" Reactor 1.XRayData[17,567,8],1"
```

This gives access to the array element XRayData[17,567,8].leakage. Using only two or three denominators in the AccessList is not allowed, since this would address a column in the matrix, resp. a structure.

EXAMPLE 3
```
typedef struct {
    double speed;
    double temperature;
} motordata;
[
    object,
    uuid(7f41e864-ba54-11d1-912c-00a02442df7d),
    oleautomation,
    dual,
    pointer_default(unique)
]
interface IMotor : IDispatch
{
    ...
    [propget] HRESULT MotorData([out, retval] SAFEARRAY *pVal);
};
```

If the MotorData above holds just a structure with type motordata, the identifier to access the “speed” sub element of the structure would be

```
" Motor 1.MotorData,0"
```

This gives access to the structure element MotorData.speed.

The access to sub elements is only supported for Base Object Version greater or equal to 2.

7.3.3.3.4.4 Quality of Service (QoS)

The Quality of Service (QoS) is defined as the maximum time duration for transmitting an item change on the provider to the consumer. The QoS is affected by communication times and communication cycles. The number of communication cycles depends on

- whether provider and / or consumer reside on Ethernet or some field bus,
- the communication channel used (local, ORPC or RT).

Parts of communication times and communication cycles are unfortunately outside control of the distributed automation FAL; this applies to

- the application cycle (also inside a field bus device),
- IP suite ASE and Ethernet medium delays, like cable length, switch jams or Ethernet controller jams,
- the field bus cycles.

Therefore the QoS covers only the times under the distributed automation FAL control, i.e. without any application or field bus cycles. Those cycle times need to be added by the user.

The QoS is defined with the following two parameters

QoS type
This parameter identifies the intended communication channel and, where applicable, a subtype; for example ORPC communication channel with subtype "productive data".

NOTE 1 There is no explicit QoS type "local". A distributed automation device decides for itself, which connections to treat locally or remotely, depending upon provider of the connection living in the distributed automation runtime instance as the consumer.

QoS Value
This parameter carries additional channel specific information. For RT, ORPC and local connections, the QoS Value parameter describes the latency, i.e. the maximum time required to propagate an item change from the provider to the consumer.

NOTE 2 This latency refers exclusively to times under distributed automation FAL control, i.e. without any application cycle, and where applicable without any field bus cycle or application cycle in a field bus device. Therefore to determine the total latency these application cycle times should be added onto the QoS Value. Since those application cycle times occur in provider and consumer, both sides should be taken into account when determining total latency.

The following QoS Types are defined:

Cyclically on changes via ORPC
This QoS type uses the ORPC communication channel. It is offered with the subtypes defined in Table 169.

Table 169 – QoS subtypes in the ORPC communication channel

Subtype	Description
productive data connections	To be used for productive data connections, implementing the distributed automation application.
status connections	To be used for diagnosis connections, allowing an engineering system and HMI application (for example an OPC Server) to display at least a few specific status variables describing the systems state even if the system is congested. These connections are scarce and are therefore only allowed to connect to system variables, e.g. to display operating and error state. This subtype is supported with Base Object Version greater or equal to 2.
HMI connections	To be used for HMI connections, allowing for example an OPC Server in a HMI role or an engineering system to display online values of items. This subtype is supported with Base Object Version greater or equal to 2.

The subtypes offer the device the possibility to group productive connections, HMI connections and status connections in both static resources and dynamic behavior. If the device is not able to fulfill all connections at some specific moment, it is able to prioritize connections along the given priority. A device has to implement the priority ranks at least for the static resources starting with Base Object Version 2, whereas implementation of the dynamic behavior is optional.

NOTE 3 Engineering systems and HMI applications are expected to use for online view purposes HMI connections only by talking to devices with Base Object version greater or equal to 2.

As for the productive operation for the connections installed with this QoS (applies to all subtypes), only changes to the item's value and quality code (see 7.3.3.3.4.8.2) of some item connected are transmitted, whereas transmission of changes to the item's value can

further be restricted by the use of the epsilon value, especially for floating point values (see 7.3.3.3.4.5).

NOTE 4 This effectively reduces network load, in particular with slowly changing data, but is only possible based upon a request response protocol.

Cyclically via Real Time (RT) protocol RT_CLASS_2
This QoS type uses the RT communication channel. It is supported with Base Object Version greater or equal to 2 and offered with the subtypes defined in Table 170.

Table 170 – QoS subtypes in the RT communication channel

Subtype	Description
RT_CLASS_2	as supported by RT (see 6.3.4 for details)

As for the productive operation for the connections installed with this QoS, the item's value and quality code (see 7.3.3.3.4.8.2) of some item connected are transmitted in every cycle.

Connection with a constant
This QoS type uses the CONSTANT communication channel. No subtype is defined for this QoS type.

For a connection with the QoS type "Connection with a Constant", the connection source is merely a constant.

The QoS Value is reserved and shall be set to zero.

The definition in Table 171 applies to the QoS type and QoS Value values that are required at the ACCO class interfaces.

Table 171 – QoS Types and Values

Quality of Service (QoS)	QoS type	QoS Value	Allowed Values
Cyclically on changes via ORPC - for productive data connections	0x00	Milliseconds	1, 2, 5, 10, 20, 50, 100, 200, 500, 1 000
Cyclically on changes via ORPC - for status connections	0x02	Milliseconds	500, 1 000
Cyclically on changes via ORPC - for HMI connections	0x03	Milliseconds	1, 2, 5, 10, 20, 50, 100, 200, 500, 1 000
Cyclically via RT - RT_CLASS_2	0x30	Milliseconds	1, 2, 5, 10, 20, 50, 100, 200, 500, 1 000
Connection with a constant	0x20	reserved	0

The QoS type 0x01 "Cyclically on changes via ORPC (sec)" is obsolete and shall not be supported in devices with Base Object Version greater or equal to 2.

With respect to the QoS, a device has subset capabilities in the following points:

- Supported QoS Types,
- Supported minimum QoS Value of a specific QoS Types.

QoS Values above a minimum QoS Value are not subset capable. If a device provides, for example, 10 ms as the lowest QoS Value it supports, it shall also support the QoS Values 20, 50, 100, 200 500 and 1 000 ms. QoS Values that are not supported are rejected with the error CBA_E_QOSVALUEUNSUPPORTED.

Further parameters describing the device-bound minimum QoS Values (according to their QoS type) are available.

NOTE 5 For all cyclic QoS Types (i.e. cyclically via ORPC or RT) the user should apply the sampling theorem if a reconstruction of a frequency is desired. This means that a QoS of at least half the sampling interval should be specified.

7.3.3.3.4.5 Dead band and epsilon value

For a QoS that defines a transmission on changes – this applies to the ORPC communication channel, an epsilon value can be optionally specified on establishing a connection via the AddConnections service. The dead band defines a new transfer that takes place only after a percentaged deviation of the value from the variable's range, which is defined through low limit and high limit.

The new value is only passed on if the absolute change of the variable exceeds this percentage. This means that the following inequality shall be fulfilled:

```
ABS(LastTransmittedValue – CurrentValue) > Deadband * (HighLimit – LowLimit)
```

This is additionally (besides transmission only upon a change) used to reduce communication on the network. It chiefly prevents „oscillations" of a floating point value from being transferred.

Since the right-hand side of the above inequality is a constant value, the epsilon value is defined in the following way. This helps to delimit the multitude of parameters to a certain extent.

```
ε = Deadband * (High_Limit – Low_Limit)
```

The epsilon value describes the absolute change of the value. To transfer the value anew, the following inequality shall therefore be fulfilled:

```
ABS(LastTransmittedValue – CurrentValue) > ε
```

The epsilon values therefore defines that a new transfer takes place only after an absolute deviation.

NOTE 1 Checking a value against the epsilon value can be omitted (even if specified) in the case of local connections. This permits a swift optimization of the connection of local data, since the epsilon check itself may already take longer than just copying the data.

An empty epsilon value is used if all data changes shall be transferred. Connections with a constant will return CBA_E_INVALIDEPSILON if an epsilon value is configured.

To specify an empty value, the VARIANT used to transmit the epsilon value through the AddConnections service shall have the data type VT_EMPTY.

An epsilon value can also not be expedient for a given connection. Table 172 defines whether epsilon values are supported for specific data types at all and what restrictions apply. If an epsilon value is used, which is not supported according to this list or which value falls into the unsupported range, CBA_E_INVALIDEPSILON will be returned.

Table 172 – Epsilon value for connectable data types

VARTYPE	Supported Epsilon Values
VT_I1	Epsilon values >= 0 are supported
VT_UI1	
VT_I2	
VT_UI2	
VT_I4	
VT_UI4	
VT_R4	Positive zero and normals are supported; denormals, negative zero and normals, positive and negative infinity, indefinite and NaN (“Not a Number”) are not supported.
VT_R8	
VT_DATE	
VT_BOOL	Epsilon value is not supported for these data types, data is transferred on every change
VT_BSTR	
VT_SAFEARRAY	
VT_USERDEFINED	
VT_RECORD	

NOTE 2 Caution needs to be used by comparing floating point values (VT_R4, VT_R8 and VT_DATE). Only if the current value and the last transmitted value are normal or zero, the above stated comparison can take place without raising a floating point exception. If any value is denormal, positive or negative infinity, NaN or indefinite the comparison is omitted and the value is transmitted, if the data changes at all – decided by a pure memory compare.

The epsilon value specified for VT_DATE is used as a date difference, meaning that the double supplied as epsilon value is interpreted as a time span. Examples for calculating an appropriate epsilon value follow below:

EXAMPLES	1 day	1.0
	1 hour	1.0/24.0
	1 minute	(1.0/24.0)/60.0
	1 second	((1.0/24.0)/60.0)/60.0

7.3.3.3.4.6 Version of a connection

To assist an engineering system on keeping the local existing connection information in an actual state, and simultaneously to reduce the information that is to be exchanged between an engineering system and the runtime, there is a version change procedure of the connections. This version procedure enables all partners to detect changes in the connections.

The ACCO ASE shall change the version for a specific ConsumerID, when

- a new connection is set up (AddConnections) under the same ConsumerID, or
- a connection is cleared (RemoveConnections, ClearConnections).

The ACCO ASE shall take precautions that ensures that the version of a newly established connection differs from the version of a connection that was established earlier under the same ConsumerID.

NOTE Since an unsigned short is used for the version, "earlier" is rather be interpreted as back in the past for a sufficiently long time. This can also be reached by keeping the versions for unassigned (i.e. established and subsequently cleared) connections persistent.

7.3.3.3.4.7 Substitute values

The substitute value (or fail safe value) provide a safe value for the application process at the information sink (consumer) in case the connection or the information source fails. The behavior differs among various application areas, mainly between factory automation (discrete production) and process automation (continuous or batch process).

Depending on the requirements during creating connections, the following basic procedures shall be supported:

– Applying a fixed substitute value ("Substitute Set").

  This procedure is mainly used in job production to define a safe state. The default substitute value used in an engineering system shall be "0".

– Holding the last known value ("Last Usable Value").

  This procedure is mainly used for the continuous / batch process. If possible, this process shall not be interrupted. Since values in process control industry change at a rather slow rate, holding the last value is the better solution.

  The last usable value used for the substitute value is always the value, which was assigned last to the property of the RT-Auto object, either through an active connection, through a write access over the RT-Auto object's process interface or through using the services WriteItems or WriteItemsQCD of the ACCO Sync interface.

A fixed substitute value of "0" can be potentially invalid for an item of data type DATE (meaning 1899-12-30 00:00), since a device can use an internal implementation range for DATE and can be unable to support this date. Therefore an engineering system shall use „2000-01-01 00:00" for DATE instead.

By using the AddConnections service the user configures the substitute value strategy for a connection when the connection is established. A VARIANT is used for transferring the substitute value. The content of the VARIANT - mainly the variant type - determines the substitute value strategy that is valid for this connection:

– Applying a fixed substitute value ("Substitute Set") by a value in VARIANT of the corresponding variant type (e.g. VT_I4 - this includes the safe "0").
– Holding the last known value ("Last Usable Value") by an empty VARIANT of the variant type VT_EMPTY.

7.3.3.3.4.8 Quality code

7.3.3.3.4.8.1 Overview

A value shall have status information about its state (such as good, bad, simulated, substitute value) that can be used by consumers for reacting and/or displaying the state of this variable.

In the Quality code concept, an additional attribute is assigned to a value which stores the quality code. The quality code consists of three portions:

– Quality contains the state of quality of data (Bad, Uncertain, Good Non Cascade, Good Cascade)
– Quality Substatus defines a set of substatus values for each quality
– Limits contains limit information for every quality and substatus (OK, low limited, high limited, constant)

7.3.3.3.4.8.2 Using quality codes

With respect to the quality codes two classes of component producers can be distinguished:

– Producers of components that take the quality code (and in most cases the time stamp, too) into account in their components. This is typically the case in the environment of process engineering. This type of component requires an applicative integration of quality control and time stamp to be provided. Based on the quality code that is transferred to the RT-Auto object, this application is able to implement a corresponding strategy. This

means it can also take a value that is preconceived in the RT-Auto object as a substitute value.

– Producers of components that do not take the quality code into account in their components. This is typically the case in the environment of production engineering. This application cannot distinguish whether a valid value or a substitute value has been applied to the component.

On the basis of these two classes, the following two implementation types of RT-Auto objects are distinguished.

Quality code aware RT-Auto ("QC-aware")
At its input side, the RT-Auto object processes the quality codes that are supplied from the outside, and creates a quality code for the properties that are provided at the outside.

Creating and processing quality codes is within the responsibility of the RT-Auto object.

The method in which the quality code is delivered to a QC-aware RT-Auto object shall be defined in a target-specific way. From the communication perspective, there is no requirement of being able to read the quality code at an interface of the RT-Auto object. It can be determined via the ACCO Sync interface (synchronous reading or writing). The ACCO object shall know, where the Quality code of the RT-Auto object comes from and/or how it can set it. This is defined in a destination-system-specific way (i.e. it is a function of the system adaptation interface).

Otherwise, this request would require a structure (value and quality code) for a property in order to be able to read or write these two values consistently. This would only permit to connect one QC-aware RT-Auto object with another one!

Quality code unaware RT-Auto ("QC-unaware")
The RT-Auto object does not know any quality code. Acting as a substitute, the ACCO object creates and processes quality codes and the RT-Auto object is provided with the values that are compatible with the transferred quality code.

Only the Quality Codes listed in Table 173 are generated and used by the distributed automation specification. They are a subset of all defined Quality Codes.

Table 173 – Quality Codes

Quality	Substatus	Value	Comment
Good (NC)	Ok	GoodNonCascOk	The value is OK.
Good (C)	local override	GoodCascLocalOverride	The value is forced.
Uncertain	last usable value	UncertainLastUsableValue	The last valid value exists. This means that the last known value is held.
Uncertain	substitute set	UncertainSubstituteSet	The value is a configured substitute value.
Uncertain	sensor conversion not accurate	UncertainSensorNotAccurate	A QOS Violation was detected in the communication channel.
Bad	out of service	BadOutOfService	The value is bad since the component is currently not processed, e.g. LDev in state CBAReady or communication fault.

The definition of the behavior for a component-overlapping quality code (see 7.3.3.3.4.8.14) has to be taken into account in order to support monitoring (i.e. the dynamic diagram or a HMI application). The quality code of an interconnected data item shall be passed on and/or can only get worse.

7.3.3.3.4.8.3 Standard behavior for quality codes

Any mixture of QC-aware and QC-unaware RT-Auto objects can exist. Therefore, the quality code is always transferred between the ACCOs as a qualifier of a variable.

Figure 21 represents the standard behavior of the Quality code transfer.

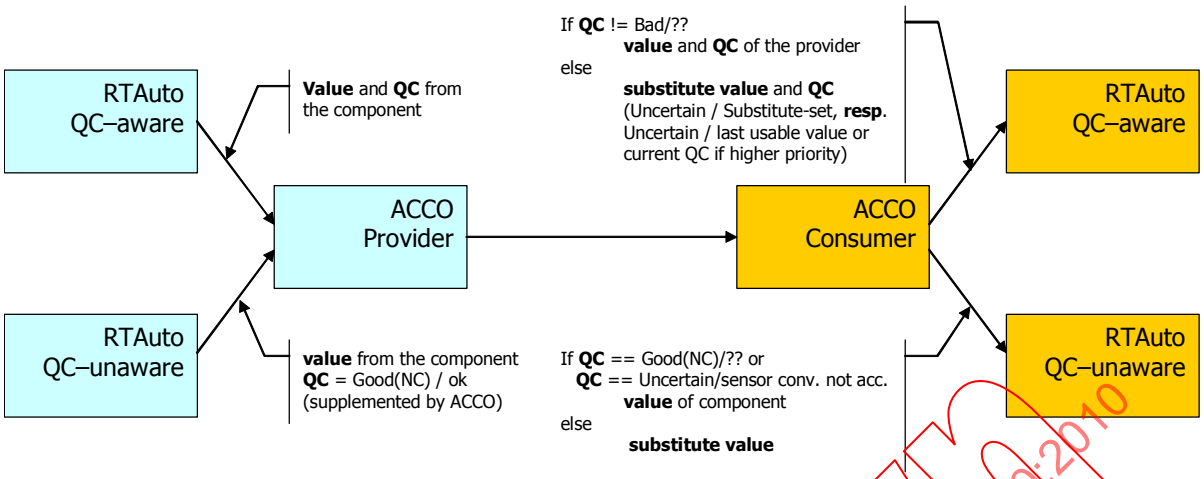


Figure 21 – Quality code transfer – standard behavior

The behavior of the provider is defined by the implementation of the RT-Auto.

– If the RT-Auto object is QC-aware, the provider accepts its Quality code but for the scenarios that are identified explicitly below.

– If the RT-Auto object is QC-unaware, the provider always sets the quality code to "Good(NC) / ok".

The behavior of the consumer is also characterized by the implementation of the RT-Auto.

– If the RT-Auto object is QC-aware, the value specified by the provider and the associated Quality code are transferred transparently to the RT-Auto object - provided that the Quality code is different than "Bad".

  Depending on the configured substitute value strategy, a Quality code "Bad" with any substatus leads to the assignment of a fixed substitute value with the Quality code "Uncertain / Substitute-set" or to holding the last known value.

  The definition of the response for a component-overlapping quality code (see 7.3.3.3.4.8.14) has to be taken into account by using the substitute strategy “Last Usable Value”. If the current quality code is of higher priority than “Uncertain / last usable value” (according to Table 174) the current quality code stays. If the current quality code is of lower or equal priority the quality code “Uncertain / last usable value” will be applied. Otherwise the quality code of the last usable value would be improved.

– If the RT-Auto object is QC-unaware, the RT-Auto object has only one value assigned. This value depends on the quality code transmitted by the provider. The transmitted value is transferred if it is "Good (NC)" with any substatus.

  Any other quality code leads to applying a fixed substitute value or to holding the last known value (depends on the configured substitute value strategy).

The difference in the behavior of QC-aware and QC-unaware RT-Auto objects is in the assessment of values with the Quality code with the status "Uncertain" or "Good(C)". Here, the ACCO object alone cannot be used for deciding whether or not they are useful. Thus, they are sent to a QC-aware RT-Auto object which can decide for itself from considering the Quality code. For a QC-unaware RT-Auto object, however, they are considered as being bad.

The only exception of this is rule is the Quality code "Uncertain/sensor conversion not accurate". Here always the transmitted value gets delivered to a QC-unaware RT-Auto object.

7.3.3.3.4.8.4 Startup of a connection

During the establishment of a connection, independent whether the connection is newly installed via the AddConnections service or restored out of the connection data base after power up, the connection sink shall obtain a defined value.

The value of the RT-Auto object during the startup is defined by the substitute value strategy, either by setting the fixed substitute value or the last known value. The latter is during power on defined by the device.

NOTE A PLC for example could use the buffered value last used before a power-off.

As shown in Figure 22 depending on the substitute value strategy, a QC-aware RT-Auto object has the quality code „Uncertain / Substitute-set" or „Uncertain / last usable value" assigned.

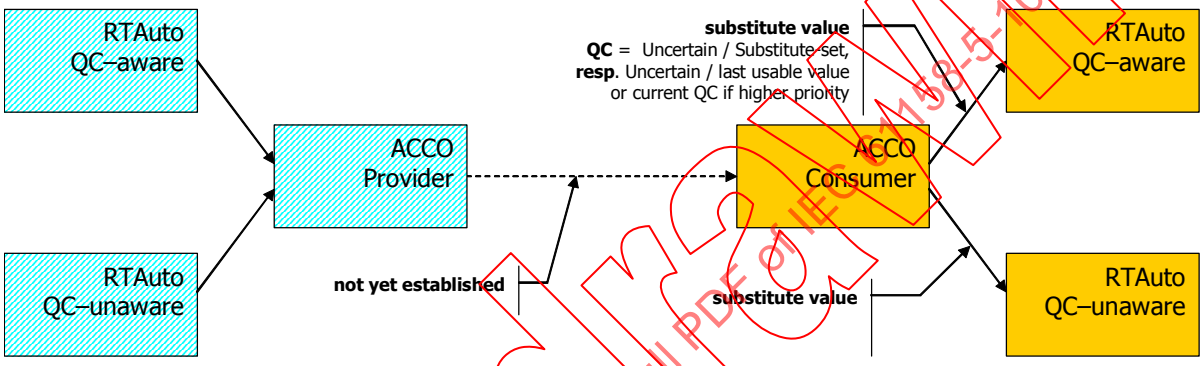


Figure 22 – Startup of a connection

7.3.3.3.4.8.5 Behavior in case of a communication fault

Irrespective of the QC-awareness of the RT-Auto object, a fixed substitute value is applied or the last value is held (depending on the substitute value strategy).

As shown in Figure 23 depending on the substitute value strategy, a QC-aware RT-Auto object has the quality code "Uncertain / Substitute-set" or "Uncertain / last usable value" assigned.

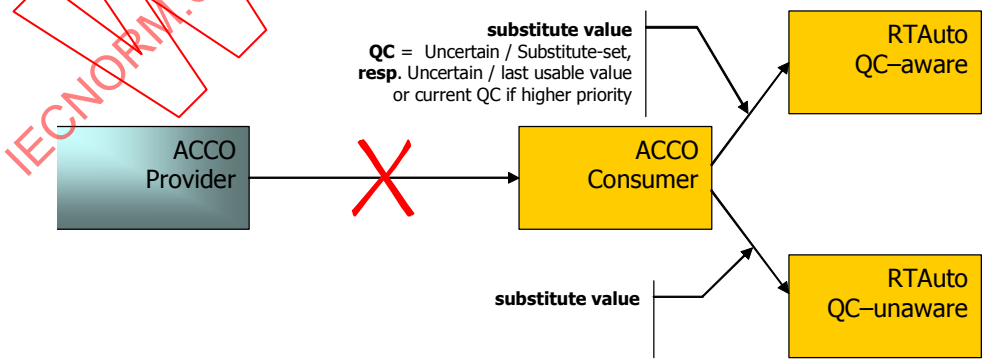


Figure 23 – Quality code with communication fault

7.3.3.3.4.8.6 Behavior in case of connection is cleared

Once a connection has been cleared, the connection sink shall obtain a defined value again. Otherwise it would hold the last transferred value and quality code. The substitute value strategy defined by the connection is used for this new initialisation.

Irrespective of the QC-awareness of the RT-Auto object, a fixed substitute value is applied or the last value is held (depending on the substitute value strategy).

As shown in Figure 24 depending on the substitute value strategy, a QC-aware RT-Auto object has the quality code "Uncertain / Substitute-set" or "Uncertain / last usable value" assigned.

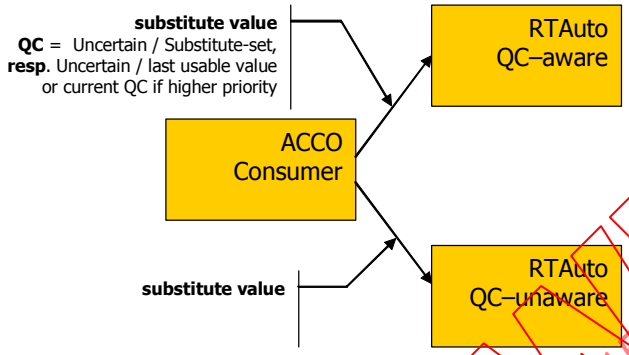


Figure 24 – Quality code when a connection is cleared

7.3.3.3.4.8.7 Behavior in case of connection is deactivated

Deactivating a connection corresponds qualitatively to clearing a connection (with holding the resources at the same time). The connection sink shall again obtain a defined value. Otherwise it would hold the last transferred value and quality code. The substitute value strategy defined by the connection is used for this new initialization.

Irrespective of the QC-awareness of the RT-Auto object, a fixed substitute value is applied or the last value is held (depending on the substitute value strategy).

As shown in Figure 25 depending on the substitute value strategy, a QC-aware RT-Auto object has the quality code "Uncertain / Substitute-set" or "Uncertain / last usable value" assigned.

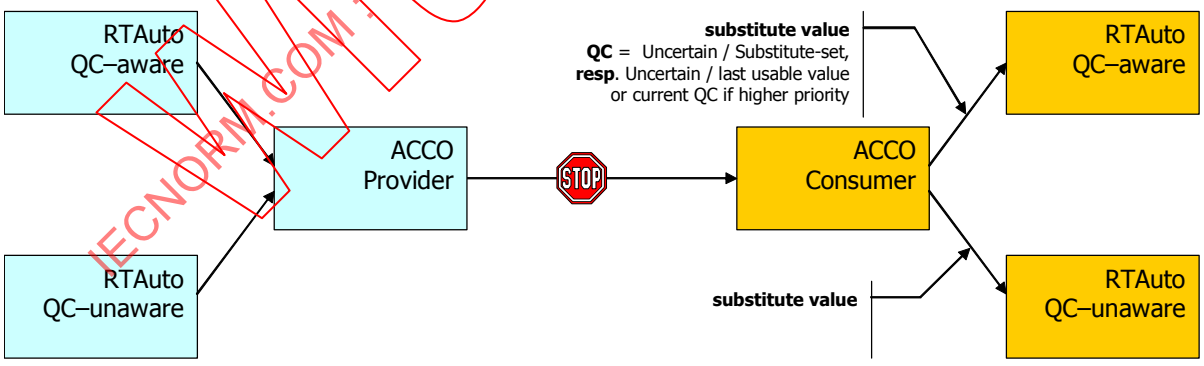


Figure 25 – Quality code when a connection is deactivated

7.3.3.3.4.8.8 Behavior in case of "incorrect" connection data

"Incorrect" connection data can occur in the following cases:

– The expected length of the connection data does not corresponds to the transferred length. This can happen, for example, in a connection between structures if the provider's structure does not corresponds to the consumer's structure (see also subclaus 7.3.3.3.4.2).

– Transfer of connection data that exceed consumer limits, like going beyond the maximum upper limit for a string or the array dimension and/or limits.

– Transfer of incorrectly serialized connection data.

Irrespective of the QC-awareness of the RT-Auto object, a fixed substitute value is applied or the last value is held (depending on the substitute value strategy). In addition, the error "type Mismatch" or "Limit violation" is logged for the connection.

As shown in Figure 26 depending on the substitute value strategy, a QC-aware RT-Auto object has the quality code "Uncertain / Substitute-set" or "Uncertain / last usable value" assigned.

NOTE For devices with Base Object Version greater or equal to 2, the first two scenarios are excluded already, when the connection was been established, since the type description is more detailed as for Base Object Version 1.

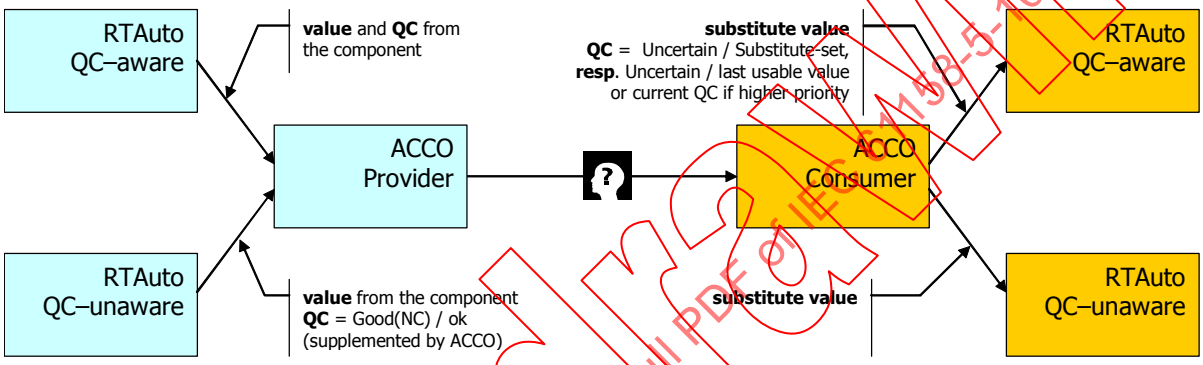


Figure 26 – Quality code during the transfer of "incorrect" connection data

7.3.3.3.4.8.9 Behavior in case of the provider in "CBAReady" state

If the provider is in the "CBAReady" state, the RT-Auto objects no longer supply valid values. Their values do not change at the side of the provider. In this situation, the provider replaces the QC of a QC-aware component and/or adds the Quality code "Bad / out of service" to the Quality code of a QC-unaware component, and transfers this code (as usual) to the consumer. This does not apply to system properties.

For the consumer, the behavior corresponds to the standard behavior described above: Irrespective of the QC-awareness of the RT-Auto object, a fixed substitute value is applied or the last value is held (depending on the substitute value strategy).

As shown in Figure 27 depending on the substitute value strategy, a QC-aware RT-Auto object has the quality code "Uncertain / Substitute-set" or "Uncertain / last usable value" assigned.

This behavior does not apply to connections with a System RT-Auto objrct as provider. All values of such RT-Auto objects remain valid even during the state “CBAReady”.

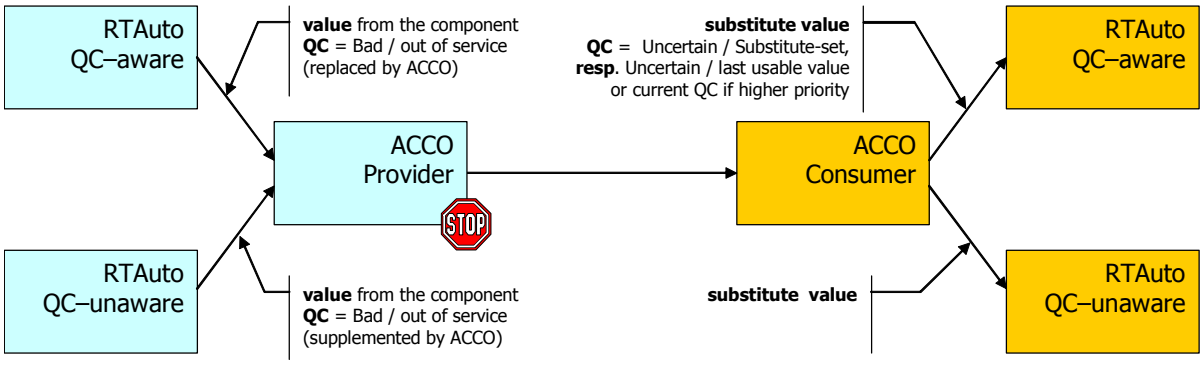


Figure 27 – Quality code for provider in "CBAReady" state

7.3.3.3.4.8.10 Behavior in case of clearing an object from the provider

If an object with an active connection is cleared from the provider, the provider transfers the last known value together with the quality code "Bad / out of service".

For the consumer, the behavior corresponds to the standard behavior described above: Irrespective of the QC-awareness of the RT-Auto object, a fixed substitute value is applied or the last value is held (depending on the substitute value strategy).

As shown in Figure 28 depending on the substitute value strategy, a QC-aware RT-Auto object has the quality code "Uncertain / Substitute-set" or "Uncertain / last usable value" assigned.

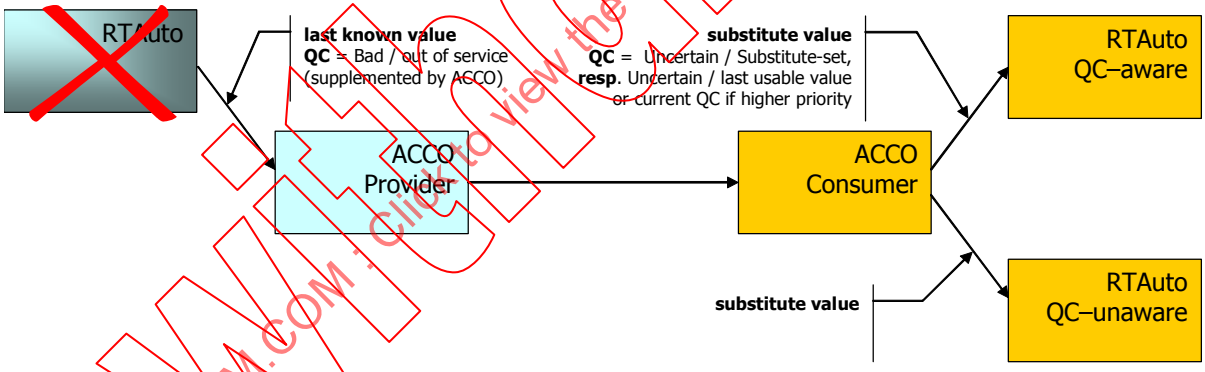


Figure 28 – Quality code when clearing an object from the provider

NOTE The provider has knowledge, when an object is cleared, either via an error code returned from the property access (CO_E_OBJECTDISCONNECTED) or by internal knowledge.

7.3.3.3.4.8.11 Behavior in case of connection is forced

Forcing a connection holds the connection sink at a fixed value that is specified from the outside.

Irrespective of the QC-awareness of the RT-Auto object, the force value is assigned instead of the transferred value.

A QC-aware RT-Auto object has additionally the "Good (C) / local override" quality code assigned. Figure 29 shows the behavior.

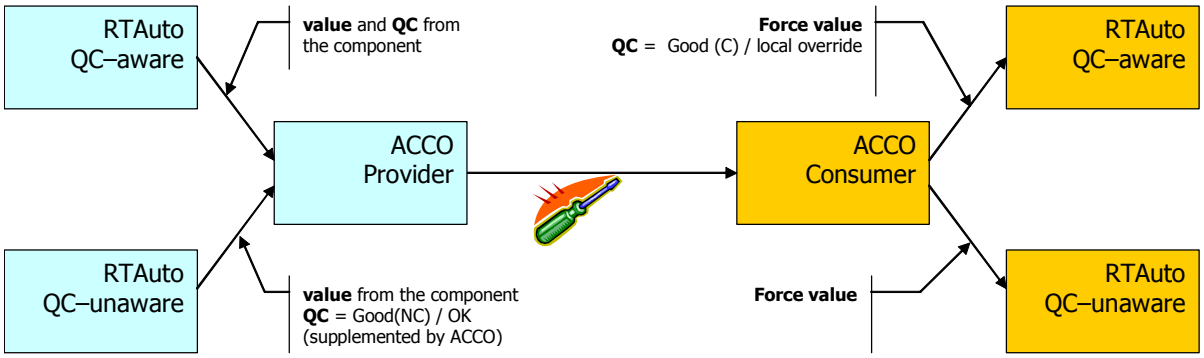


Figure 29 – Quality code when a connection is forced

Forcing a connection is not supported for Base Object Version.Major equal to 1 or 2.

7.3.3.3.4.8.12 Behavior in case of QoS violation

If the provider is no longer able to maintain a Quality of Service for a connection (i.e. if it is no longer able to maintain the requested time intervals due to a temporary or permanent work overload), the values can no longer be provided at the required quality. In this situation, the provider replaces the QC of a QC-aware component and/or adds the Quality code "Uncertain / sensor conversion not accurate" to the Quality code of a QC-unaware component, and transfers this code (as usual) to the consumer.

If a QC-aware component already contains a Quality code that is - according to the priority Table 174 - worse than "Uncertain / sensor conversion not accurate", the Quality code that is specified by the component side shall be retained and transferred.

For the consumer, the behavior corresponds to the standard behavior described in Figure 30. Resulting from the transmitted quality code, a QC-unaware RT-Auto object gets the transmitted value without any notification of the problem. A QC-aware RT-Auto object has the transferred value with the corresponding quality code assigned. Therefore only a QC-aware RT-Auto object is capable to detect this situation.

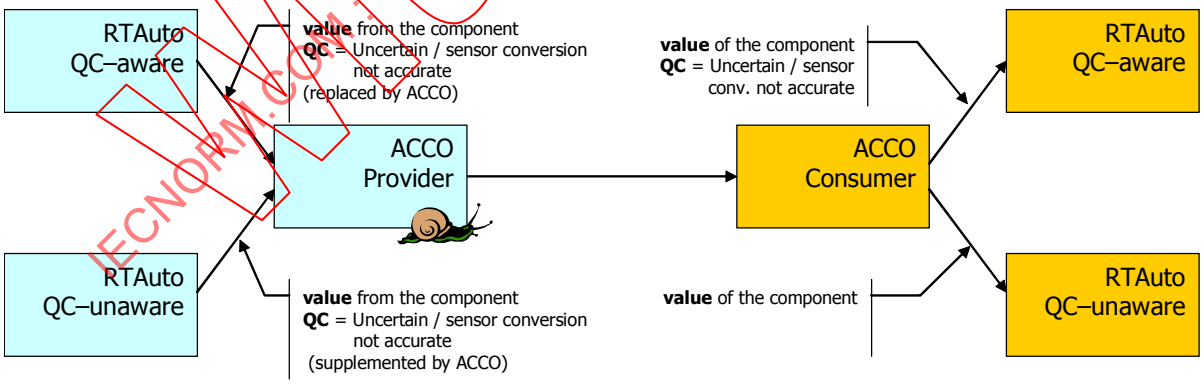


Figure 30 – Quality code at QoS violation

7.3.3.3.4.8.13 Write access to values via put-services or WriteItem services

If RT-Auto object values are written via property put on Custom RT-Auto interfaces or the ACCO service WriteItem of the ACCO Sync interface, the Quality code "Good(NC) / ok" will always be added.

7.3.3.3.4.8.14 Component-overlapping quality code

At an exact point of time, only one quality code can exist at a variable. This quality code contains exactly the state of the variable at this moment. The quality code for a variable could be potentially generated at several points. A variable in a device obtains a quality code, which results from the verification of the state of the variable in the device (a substitute value is passed on, for example, if a broken wire is detected). Then the variable will be processed furthermore. A quality code for the variable is generated there based on specific verifications (does, for example, the variable exceed a configured upper limit?).

In general it can be said that a quality code for a variable can be produced in any component in which variables are produced and/or processed. A rule is therefore required that says how the quality code is handled across the components:

> In the processing hierarchy, the quality code may only deteriorate, never improve. Each subsequent component that produces quality code determines the component-specific quality code according to its rules. Next, a priority table (see Table 174) is used for checking whether this component-specific quality code is better or worse than the one that already exists at the variable.

NOTE A component that produces variables can, of course, improve the quality code (from bad to good, for example). A subsequent component is a component that processes existing variables and their quality codes.

Two cases are possible with subsequent components:

– The component-specific quality code is better than or equal to the one that exists at the variable: Leave the quality code at the variable unchanged.
– The component-specific quality code is worse than the one that exists at the variable: Overwrite the quality code at the variable with the component-specific one.

This ensures that the quality code at the variable always maintains the worse state, irrespective of number and kind of the components in which the variable was processed.

The priority table has been defined in Table 174.

Table 174 – Quality code priority table

Quality	Substatus
Good (NC)	OK
Good (NC)	Maintenance Required
Good (NC)	Active Update Event
Good (NC)	Active Advisory Alarm
Good (NC)	Active Critical Alarm
Good (NC)	Unack Udate Event
Good (NC)	Unack Advisory Alarm
Good (NC)	Unack Critical Alarm
Good (C)	OK
Good (C)	Initialization Acknowledge
Good (C)	Initialization Request
Good (C)	Not invited
Good (C)	Do Not Select
Good (C)	Local override
Good (C)	Initiate fail safe
Uncertain	Non-specific
Uncertain	Last Usable Value (LUV)
Uncertain	Substitute
Uncertain	Initial Value
Uncertain	Sensor Conversion not Accurate
Uncertain	Engineering Unit Violation
Uncertain	Sub-Normal
Uncertain	Configuration Error
Uncertain	Sensor Calibration
Uncertain	Simulated Value
Bad	non-specific
Bad	Configuration Error
Bad	Not Connected
Bad	Sensor Failure
Bad	Device Failure
Bad	No Communication with last usable value
Bad	No Communication with no last usable value
Bad	Out of Service

The quality code of the lowest priority (the best quality code) is at the top of the table. The one with the highest priority (the worst) is at the bottom.

7.3.3.3.4.8.15 Property access and quality code

If the provider accesses an RT-Auto object to get properties, the following rules apply to form the accompanying quality code:

- If the LDev object is in state “CBAReady” or the property access service returns any ERROR code (E_*, CBA_E_*), the quality code “Bad/out of service” is delivered.
- If the property access service returns a SUCCESS code (S_*, CBA_S_*) the existing quality code will be delivered. This quality code is initialized with „Good (NC)/ok”.
- If the LDev object is in state “CBAReady” a property access returns the existing quality code on System RT-Auto objects and quality code “Bad/out of service” on Custom RT-Auto objects.

The actual return code of the property write access by the consumer is returned also in the detailed diagnosis of the ACCO object (i.e. in the state of connection), if an error code is returned. The error code of the consumer’s write access will never be returned back to the provider as a result of the OnDataChanged service resp. GetConnectionData service.

If a full-access property is a consumer for some connection and a provider for a second connection, the quality code transmitted along the second connection shall not improve the

quality code transmitted along the first connection (quality code propagation – see 7.3.3.3.4.8.14).

The same rules apply to the quality code delivered with the ReadItems service.

If a single element inside a structure or array is accessed, the following rules apply to from the accompanying quality code:

- If the ReadItems service is used, the quality code delivered will be the quality code of the item.
- If the WriteItems service is used, the quality code of the item will not be changed.
  NOTE Since the WriteItems service would always supply “Good (NC) / ok”, combining it with the rule, that the quality code shall not be improved, results in keeping the actual quality code.
- Using the WriteItemsQCD service is not allowed for single element access.

7.3.3.3.4.9 Operating state

An ACCO object is independent of the operating state. It transfers and receives connections in every operating state.

7.3.3.3.4.10 Power-On

The consumer shall re-establish all connections after a power-off. The consumer first invokes the DisconnectMe service of the ACCO Server interface to all persistent configured providers. The provider shall then discard all information that is related to this consumer if it has not yet detected the power-off state of the consumer.

The consumer shall then re-establish all connections based on its retentive data by invoking the Connect2 service of the ACCO Server interface.

The provider remains passive after a power-off. The consumer shall re-establish the connections. This means that the provider does not provide any retentiveness.

7.3.3.3.4.11 Persistence

The restart without an explicit intervention of the client (e.g. engineering system) shall be supported by the automation device. Hence, the consumer as part of the ACCO ASE shall maintain the connection information as non volatile when power is switched off. The Physical Device class provides the Save service within the ICBAPersist interface to request the persistent save (e.g. in flash memory or EEPROM) for all connection information of the whole automation device, independent, which client was adding the connection before. There is no concept of an initial state of a connection, always the current active information is saved.

If the communication to the automation device was interrupted during the connection download, the client (e.g. engineering system) shall reevaluate all connections when the communication is re-established.

NOTE 1 The automation device could be switched off during persist of the connection data base. The state of the connection data base after power-on during such an interruption is undefined. The engineering system should delete the complete connection data base after a communication interruption.

The usage model for persistence is to make all changes for all LDev objects residing on one PDev object and then to save the changes with one invocation of the Save service.

To support this model, each connection carries an identification, whether the connection is currently backed up in the persistence. This information can be fetched with the GetConnections service. The Persistence information has three states:

– CBAVolatile – the connection will never be persistent.

– CBAPendingPersistent – the active information of the connection is not equivalent to the information in the persistence, i.e. a call to the Save service of the Persist interface is needed to bring them in sync.

– CBAPersistent – the active information of the connection is equivalent to its information in the persistence.

If a connection is established via the AddConnections service of the ACCO Management interface, all connections which where requested to be persistent, carry the information CBAPendingPersistent until the Save service of the Persist interface stores the current state into the persistence data base. Then all connections are set to CBAPersistent. The next change to a connection via the SetActivationState service of the ACCO Management interface brings the connection back to the state CBAPendingPersistent.

If an error appears during the Save service, all persistent connections of the currently stored LDev object will be set to CBAPendingPersist and the configuration data base will be set to invalid. This leads to an error returned by the Save service, nevertheless the save continues for all other LDev objects. The client saving the connections is able to detect the failed connections via CBAPendingPersistent, he is forced to retry the operation (whereas the error is likely to reappear).

All [in] parameters of the AddConnections service are to be kept persistent. The ConsumerID is not kept persistent.

NOTE 2 To support performance characteristics the connections stored "know" whether they have been established as local or remote connections. Therefore if the local IP address changed for the device with connections already established, connections which have been established previously as local connections (i.e. carrying the "old" IP address as provider) would now be remote connections and those being previously remote (i.e. connections carrying the "new" IP address as provider) would be now local connections. The reestablishment could hurt the performance characteristics parameters. Therefore those connections appear neither in the consumer nor in the provider. Errors therefore may only be reported proprietarily. Those connections will be automatically cleared with the next invocation of the Save service. The same behavior – automatic clean of connections – applies to RT connections being persistent. If the restore of the connections detects that the RT protocol is no longer functional on the Ethernet interface (e.g. RT driver is inactive or deinstalled), those connections will be automatically cleared with the next invocation of the Save service. Again the error may only be reported proprietarily. This may appear typically only on PC devices, since embedded devices will not allow reconfiguring Ethernet interfaces. Components of an Ethernet interface are shown in Annex B.

This requires the following connection information to be retentive:

- Currently valid ping factor (see 7.3.3.3.5.2.5.1),
- ACCO stamp (see 7.3.3.3.4.12),
- for each connection
  - Consumer ID (see 7.3.3.3.6.2.1),
  - Identifier for the provider's LDev object (see 7.3.3.3.4.3.1),
  - Identifier for the connection source
  - Current status (active or inactive) (see 7.3.3.3.6.5),
  - type and qualifier of the QoS (see 7.3.3.3.4.4),
  - Substitute value (see 7.3.3.3.4.7),
  - Epsilon (see 7.3.3.3.4.5),
  - Version number (see 7.3.3.3.4.6).

NOTE 3 The number of connections a consumer/provider pair is able to establish mainly depends on the memory space available in the target platform. Since it can be assumed, that the non volatile memory space available is usually smaller than the memory space that is managed via the dynamic memory management, the bottleneck is with the consumer.

NOTE 4 In order to permit a swift replacement of a defective module, the persistence information should be stored on a medium that can be swapped when the defective module is replaced (e.g. an EPROM card).

Simultaneous modifying access to the connections is not possible as long as the Save service is outstanding and processed by the server. Therefore, the server shall negative respond

incoming AddConnections, RemoveConnections, ClearConnections, and SetActivationState services with CBA_E_PERSISTRUNNING error during this phase. The client application that wishes to install connections shall repeat the setup attempt after a short delay. This means that this error shall only be returned to the user after several unsuccessful attempts have been made.

NOTE 5 An LDev object will not be seen “in public” until all connections have been restored from persistence. Therefore an engineering system can assume differences between online and offline, if it browses the LDev object.

7.3.3.3.4.12 Online/offline comparison of the connections

There is a procedure defined for the online / offline comparison of the connections.

The ACCO stamp reflects changes regarding the connections the ACCO object is a consumer for and the Ping Factor. The stamp reflects both persistent and non-persistent changes.

The ACCO stamp is supported through the STAMPCOLLECTION attribute of the System RT-Auto object. It consists of three parts:

(1) The first part is altered on every service invocation of the ACCO Management interface which changes the persistent part of the connection data base (services AddConnections, RemoveConnections, ClearConnections, SetActivationState and put_PingFactor). This part is stored on persisting the connection data base via the Save service of the Persist interface and retrieved on Power-On.

NOTE 1 The change to the ACCO stamp is made on the service invocation, although the change to the connection data base is only persistent after calling the Save service. Therefore connections in state CBAPersistent and CBAPendingPersist are treated together.

(2) The second part is altered on every service invocation of the ACCO Management interface which changes the volatile part of the connection data base. This part is initialized randomly to a value other than 0.

(3) The third part is incremented on every service invocation of the ACCO Management interface which changes the connection data base. This part is called modification counter and initialized with 0 on Power-On.

NOTE 2 This three part division of the ACCO stamp ensures that any differences are well recognized. Without an individual part for non-persistent connections an erroneous equity would be signaled after re-start, although all non-persistent connections would go away by this restart.

Since every service which changes the connection data base can potentially alter both persistent and volatile information with one invocation, the ACCO stamp may alter in all parts upon one service invocation.

NOTE 3 The engineering system establishes a connection to the ACCO stamp via the Connect2 service of the ACCO Server interface and stores a local copy of the first transmitted value. All changes to the ACCO stamp will be transmitted through this connection. If the engineering system makes changes to the connection data base, it expects the modification counter to alter to a specific value by adding the number of service invocations for the changes. Therefore an engineering system is able to compute the expected modification counter of the ACCO stamp in advance, by just adding the number of service invocations needed to the start value of the modification counter. The engineering system can detect changes to the connection data base made by concurrent engineering by the following means: a) If a change to the ACCO stamp is transmitted upon changes made by the engineering system, the engineering system has to check, whether the modification counter is lower or equal to the expected value. b) If the modification counter is higher than expected, the engineering system has to compare all its connections by using the services GetConsIDs and the GetConsConnections service. c) If a change to the ACCO stamp is transmitted without any changes made by the engineering system, the engineering system has to compare all its connections by using the services GetConsIDs and GetConsConnections.

7.3.3.3.5 Communication channels

7.3.3.3.5.1 Configuration

The ORPC communication channel and the RT communication channel are defined as a common strategy for connections, where ORPC is used for sending changes and RT for cyclic transmission with realtime capabilities.

The user always configures the transmission channel (ORPC or RT) for a connection, as well as the latency time, that is the maximum time required for a change to an item of data to be transported from the provider to the consumer. This latency time does not account for application cycle times or field bus cycle times.

The user can choose from the fixed (equal) grid of latency times for ORPC and RT (1, 2, 5, 10, 20, 50, 100, 200, 500 and 1 000 ms). It is necessary to define on a product-specific basis whether a global default can be used for configuring connections (e.g. RT / 20 ms) and how the possible QoS Values will be presented to the user.

The “substitute value apply time on communication problems” can be calculated offline from the QoS Value and shall be visible to the user in a connection tool (properties of a connection). The calculation is based on the following:

- for ORPC QoS * Ping Factor * 2 => QoS * 20
- for RT 4 * RT cycle time

The “typical” transmission time is always less than or equal to the latency, and the maximum transmission time shall be somewhere between the latency and the substitute value apply time.

NOTE The apply time stated is a theoretical time, since it does not include internal application times to propagate the substitute value to the user’s application. Accordingly these two times do not have to be made available to a user.

7.3.3.3.5.2 ORPC communication channel

7.3.3.3.5.2.1 General

The ORPC communication channel is mandatory. It enables connections to be transmitted via the ORPC protocol.

7.3.3.3.5.2.2 Context management

7.3.3.3.5.2.2.1 Overview

The ACCO Server interface (the Connect2 service in particular) is used as context management for the ORPC communication channel. The communication channel is established by transferring the Callback Interface Pointer at the Connect method.

The Connect2 service supports two operation modes of the communication channel, the so-called push mode and the pull mode. A consumer has to decide which operation mode to use. Changing the operation mode of a channel while still connections having installed to a consumer is not supported and will be denied with CBA_E_MODECHANGE.

While the consumer has to provide a callback interface for using the push mode, the pull mode works completely consumer-driven.

NOTE The pull mode enables security aware clients having a firewall installed to still get connection data without need for parameterizing the firewall.

Both operation modes are working change driven, i.e. connection data is only transferred if values along a connection have changed. In push mode such values are pushed actively by the provider, whereas in pull mode the connection data is transferred as a response to a consumer’s request.

7.3.3.3.5.2.2.2 Push mode

Using push mode a provider pushes the connection data actively to a consumer via the OnDataChanged service of the ACCO Callback interface.

Figure 31 gives an overview over the responsibilities of both consumer and provider regarding push mode. The consumer is responsible to establish the communication via the Connect2 service of the ACCO Server interface. The provider is responsible to push the data actively via the OnDataChanged service of the ACCO Callback interface.

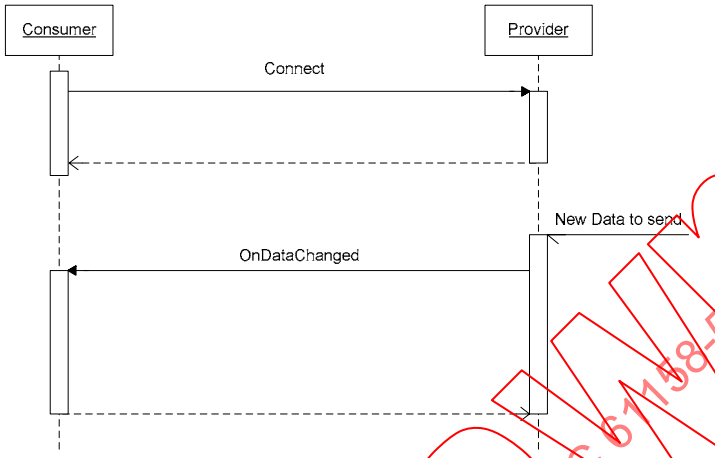


Figure 31 – Push mode

If different interface instances of the callback pointer are transmitted upon repeated invocations of the Connect2 sevice by the same consumer, the provider only employs the instance last transmitted. Thus it is able to automatically group connection data and transmit it in an optimized way.

A provider shall not invoke the OnDataChanged service significant faster than the configured QoS for not to overload the consumer and the network. Since the service invocations to a single consumer will be serializes and the single invocation is timeouted, the consumer is expected to response as fast as possible if invoked via the OnDataChanged service.

7.3.3.3.5.2.2.3 Pull mode

Using pull mode a consumer pulls the connection data from a provider via the GetConnectionData service of the ACCO Server interface. The pull mode is supported for Base Object Version greater or equal to 2.2.

Figure 32 gives an overview over the responsibilities of both consumer and provider regarding pull mode. The consumer is responsible to both establish the communication via the Connect2 service and to get the data via the GetConnectionData service of the ACCO Server interface.

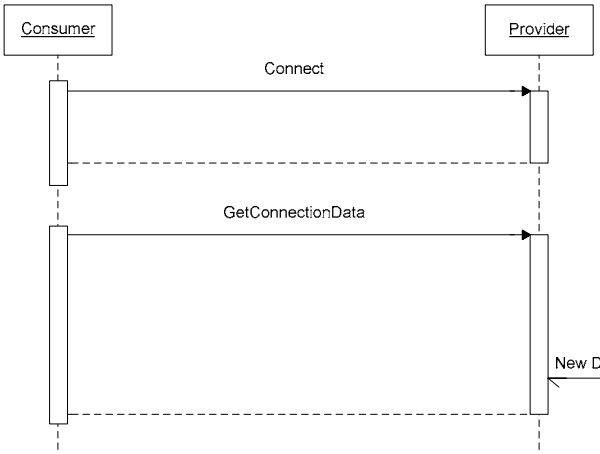


Figure 32 – Pull mode overview

Figure 33 gives a detailed overview regarding pull mode.

Pull mode does not mean polling, since the GetConnectionData service request is held until new changed data is available, resp. a maximum hold time has exceeded.

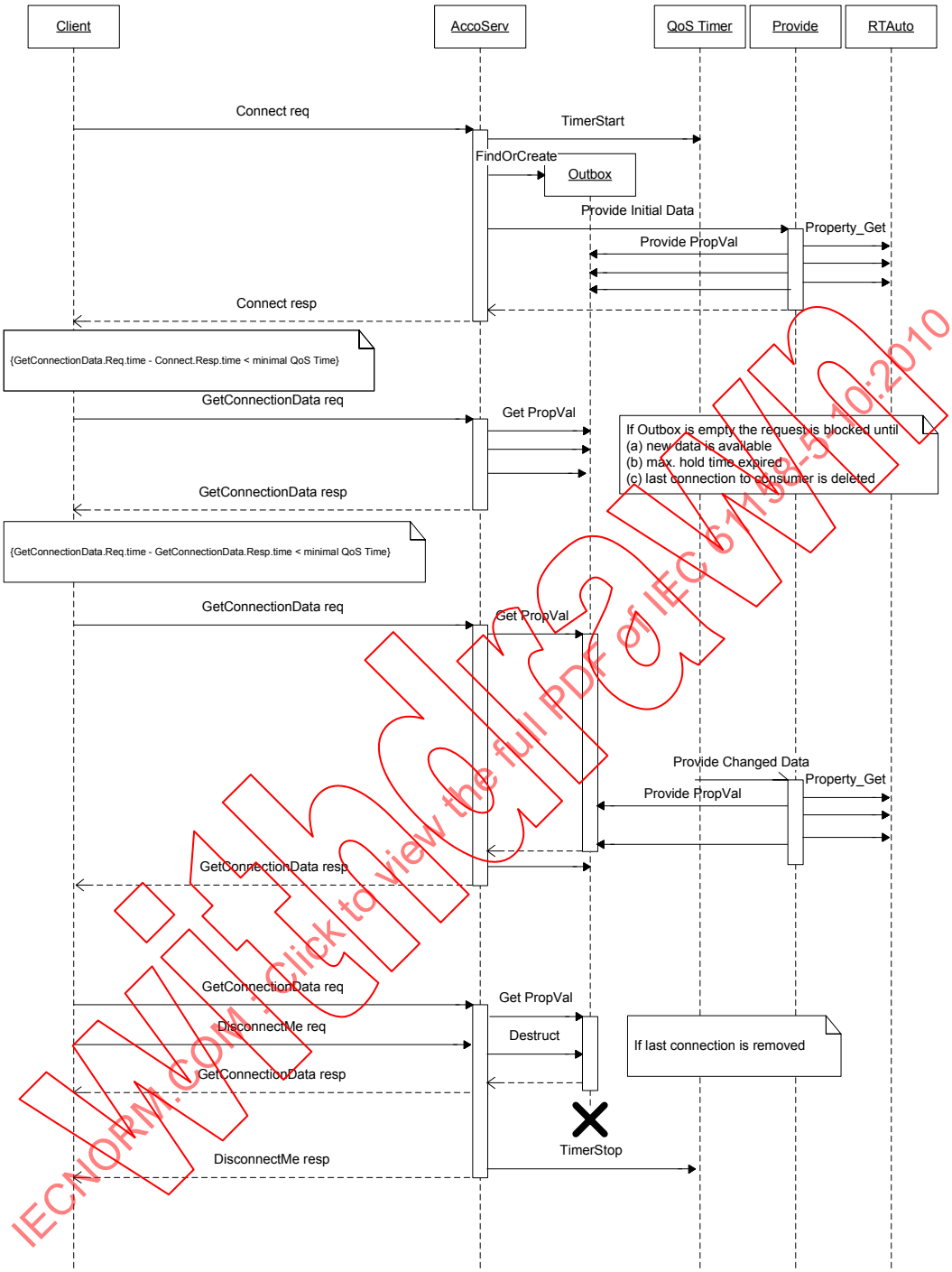


Figure 33 – Detailed sequence chart of the pull mode

7.3.3.3.5.2.3 QoS

In the ORPC communication channel the QoS is fulfilled by the following means:

The sampling rate of the ACCO provider cycle is shorter than the QoS Value. The time measured between start of the sampling, collect values from the application and the OnDataChanged service invocation (i.e. transferring all changed values to the consumer, unpack the buffer and deliver the values to the application) including return back to the consumer is always shorter than the QoS Value.

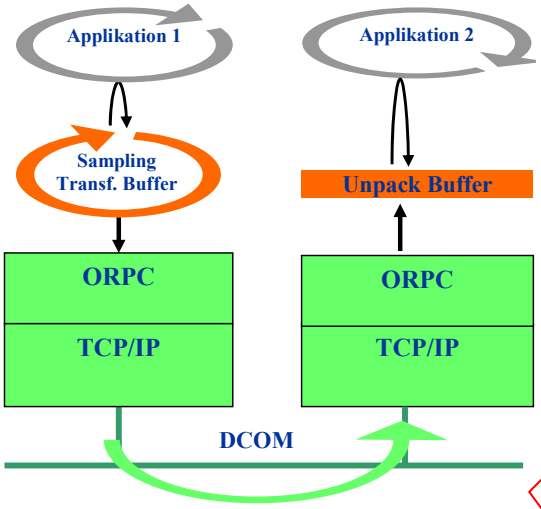


Figure 34 – QoS and ORPC communication channel

7.3.3.3.5.2.4 QoS violation

For data connections the provider has to check, whether a QoS violation happened and takes appropriate actions to propagate the QoS violation to the consumer by setting the quality code of the affected connections.

There are two reasons for a QoS violation:

(1) The device is not able to hold the sampling rate, i.e. the time delta between the last and the current sampling is higher than the sampling rate given by the user.

(2) A new sampling is triggered before the OnDataChanged service confirmation of the last sampling is received resp. a new GetConnectionData service invocation was issued.

In both cases all connections to this consumer with the same sampling rate (derived from the QoS Value) are transmitted with the quality code “Uncertain / sensor conversion not accurate”, with keeping the quality code propagation in mind as not to improve the quality code (see 7.3.3.3.4.8.14). All values with a new quality code are transmitted in case (1) in this very sampling, in case (2) when the pending OnDataChanged confirmation is received, resp. the new GetConnectionData request is issued.

The QoS violation is canceled by transmitting any other quality code than “Uncertain / sensor conversion not accurate” for the affected connections.

For Base Object Version less than 2 the QoS violation is canceled with the next sampling, where the situation is back to normal, i.e. the time delta being back to lower than the sampling rate and the transmission having already finished when the new sampling starts. This may lead in an overload situation to a toggling between QoS violation indicated and back to a normal, since again all affected connections are transmitted within one sampling, thus making case (1) more likely for the sampling following.

For Base Object Version greater or equal to 2 the QoS violation will be canceled more smoothly to decrease the chance of toggling. If a QoS violation occurs, a count is remembered for all affected connections, which indicates after what number of samplings the QoS violation is terminated at the latest by transmitting the current quality code (different from “Uncertain / sensor conversion not accurate”). The count is set to 8, 9, and 10 for every third of the affected connections, to spread out the load on canceling the QoS violation.

On every sampling, the count of the affected items is decremented. If in a sampling the count is decremented to zero, the item will be transmitted with its actual value and quality code.

Whenever an item is transmitted because its value or quality code changed, the count is reset to zero, thus avoiding an explicit transmission to cancel the QoS violation. If during the period another QoS violation occurs, only affected connections with count equal to zero will result in a transmission with “Uncertain / sensor conversion not accurate” if not having a worse quality code. The affected connections with count unequal to zero have already been transmitted with the last QoS violation. All affected connections will then again be setup with 8, 9 and 10 for the count as stated already above.

This mechanism spreads out the load for canceling the QoS violation effectively.

Figure 35 shows a sequence chart for the behavior within pull mode.

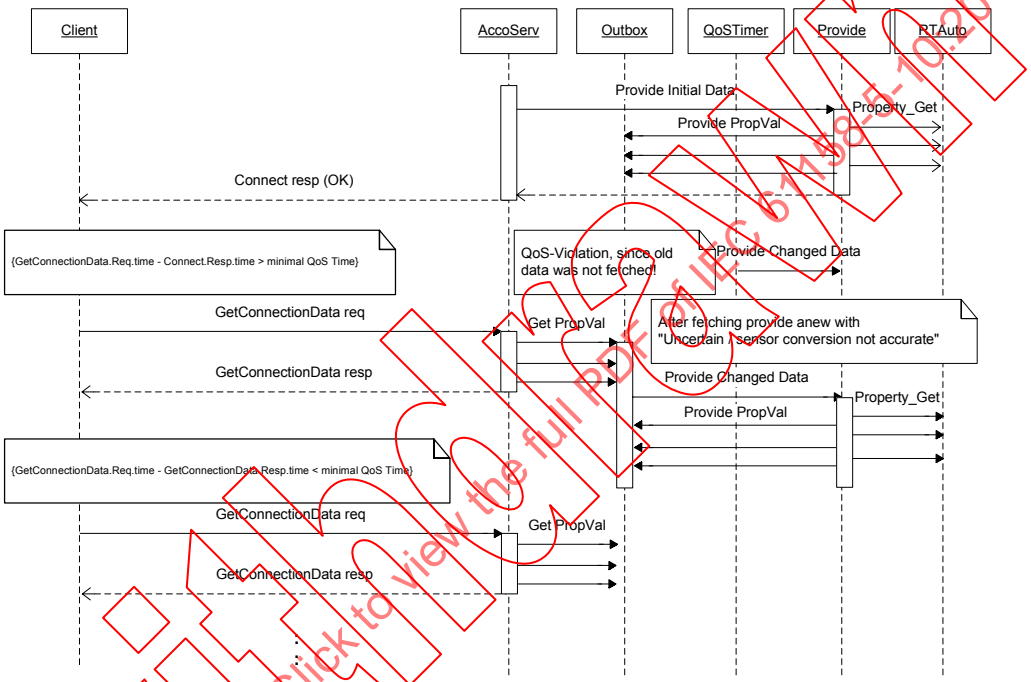


Figure 35 – QoS Violation within Pull Mode

7.3.3.3.5.2.5 Connection monitoring

7.3.3.3.5.2.5.1 Connection monitoring by the consumer – Push mode

For push mode the provider provides the Ping service of the ACCO Server interface for monitoring the provider's heartbeat. This ping checks all connections that this provider delivers to the consumer with one call.

The consumer shall invoke this method if it has not heard anything from a given provider after a certain dead time (i.e. if that provider has not supplied any data to the consumer during this interval), independent of the connection state (active or inactive) or the consumer’s operating state (CBAReady or CBAOperating); even if the consumer knows, that no data will arrive. This is done to keep the connections established alive, since if the provider detects unilateral a communication failure, he would drop the connections.

To reduce network load, the consumer doesn’t ping, if the provider delivers data often enough to the consumer; the provider’s invocation of the OnDataChanged service is used as an implicit heartbeat.

A client (e.g. a engineering system) configures this dead time via the put_PingFactor service of the ACCO Management interface. It is specified as a factor. A consumer can tolerate silence of a provider up to the n-fold dead time of the minimum Quality of Service of all connections delivered by this provider, until he checks the availability of the provider, using the Ping service of the ACCO Server interface. If the provider is dead, this is indicated by an HRESULT different than S_OK; the consumer applies substitute values on the connection sinks.

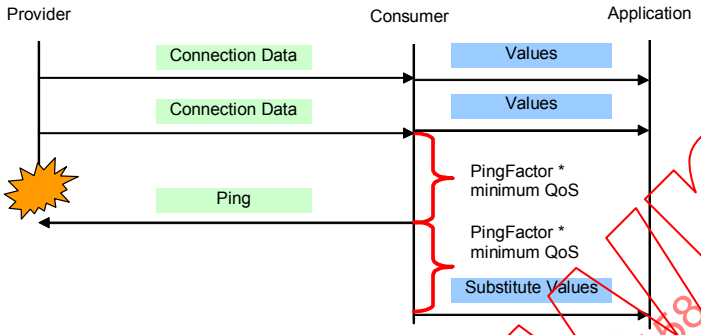


Figure 36 – Monitoring the providers heartbeat

To provide a better timeout behavior and in consequence a faster apply of substitute values, the provider is considered already dead, if the Ping service confirmation is not received after the n-fold minimal QoS.

Therefore the maximum time between the death of the provider and apply of the substitute value is

2 * (PingFactor * MinQoS).

The following definitions are made for the ping factor:

- The default value is 10.
- Setting the ping factor to 0 disables pinging.

Table 175 shows the maximum apply time of substitute values in case of communication errors, whereas the ping factor is assumed on its default value of 10.

Table 175 – Maximum ORPC substitute value apply time

QoS Value	Max. substitute value apply time (in ms)
1	**20**
2	**40**
5	**100**
10	**200**
20	**400**
50	**1 000**
100	**2 000**
200	**4 000**
500	**10 000**
1 000	**20 000**

7.3.3.3.5.2.5.2 Connection monitoring by the consumer – Pull mode

In pull mode the consumer does not need an explicit ping as in push mode.

The provider holds an GetConnectionData service request until one of the following conditions arises:

- A QoS sampling and check for changes results in new data to be transmitted.
- There is still data to be transferred from the last QoS sampling.
- The maximum hold time expires according to Table 176; following the equation 10 * MinQoS.
- The last connection to this consumer was deleted.

Table 176 – Maximum GetConnectionData hold time

QoS Value	Max. hold time (in ms)
1	10
2	20
5	50
10	100
20	200
50	500
100	1 000
200	2 000
500	5 000
1 000	10 000

If the maximum hold time expires an empty response will be returned. If a data change is detected no GetConnectionData is hold by the provider, the connection data is queued, i.e. property values will be kept, until the next GetConnectionData request is issued by the consumer. Further value changes are unaccounted until the queued values are transmitted. Intermediate values are lost (same as for push mode).

Therefore instead of pinging the consumer has to check, whether at least all 10 seconds the provider responses to the consumer's GetConnectionData service request. Thus the confirmation acts as an implicit ping.

If the consumer receives the GetConnectionData confirmation, the consumer is advised to issue another request as soon as possible.

7.3.3.3.5.2.5.3 Connection monitoring by the provider – Push mode

The provider does not require any additional procedure to monitor the consumer's heartbeat. A provider is able to detect the consumer's failure synchronously, since a necessary data transfer to the consumer cannot take place.

The invocation of the OnDataChanged service is denied with RPC_E_* or something similar. The provider drops all connections silently, if the call fails with any error code.

7.3.3.3.5.2.5.4 Connection monitoring by the provider – Pull mode

A consumer dropping out after invoking the Connect2 service and before invoking the GetConnectionData service of the ACCO Server interface the first time or taking to long after invoking the GetConnectionData service anew is detected by a timeout. If the timer expires before a new GetConnectionData service is issued, a failure of the consumer is assumed. This failure of the consumer is treated by dropping all connections by the provider.

7.3.3.3.5.2.6 State machines

7.3.3.3.5.2.6.1 State machine RemoteACCO

Figure 37 shows the RemoteACCO state machine.

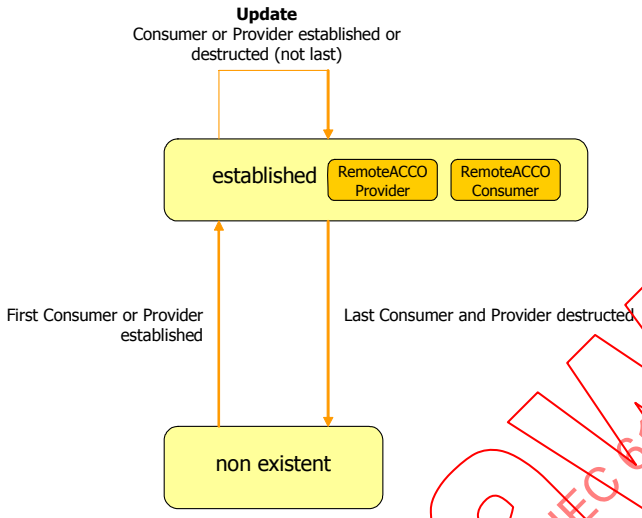


Figure 37 – State machine RemoteACCO

All connection communication is established over the RemoteACCO (Context Management). In the state “established” independent state machines for consumer and provider are running.

7.3.3.3.5.2.6.2 State machine RemoteACCOProvider

Figure 38 shows the RemoteACCOProvider state machine.

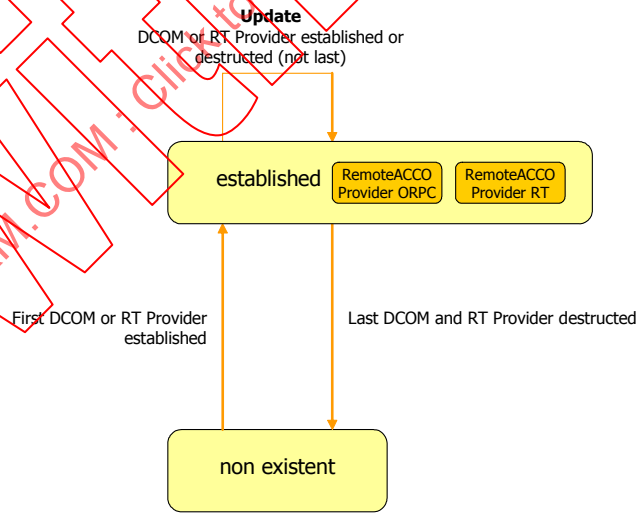


Figure 38 – State machine RemoteACCOProvider

The RemoteACCOProvider is further decomposed in two independent and reactionless parts for ORPC and RT channels.

7.3.3.3.5.2.6.3 State machine RemoteACCOProvider$_{ORPC}$

Figure 39 shows the RemoteACCOProvider$_{ORPC}$ state machine.

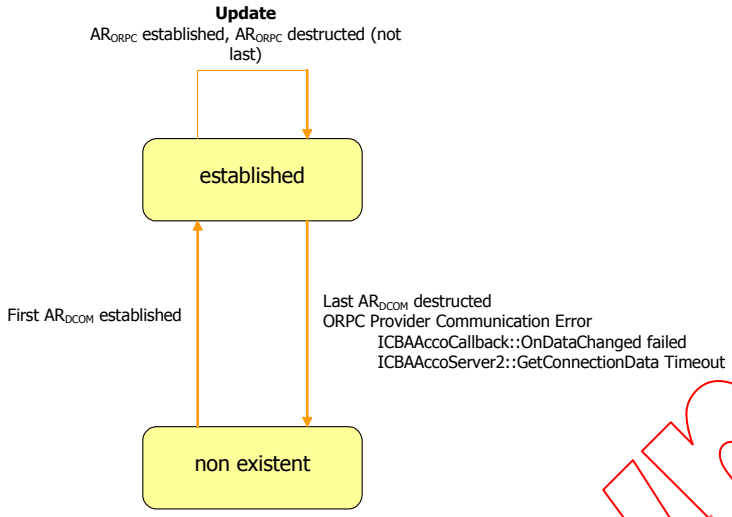


Figure 39 – State machine RemoteACCOProvider$_{ORPC}$

A RemoteACCOProvider$_{ORPC}$ collects all application relations (AR) related to the same consumer ACCO object, therefore all ACCO objects of a provider application regarding ORPC channels (productive, HMI and status connections).

7.3.3.3.5.2.6.4 State machine AR$_{ORPC}$

Figure 40 shows the AR$_{ORPC}$ state machine.

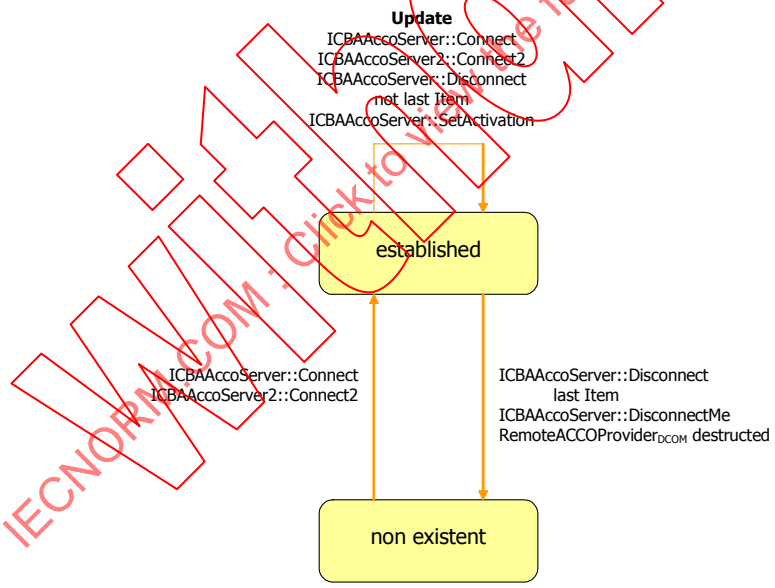


Figure 40 – State machine AR$_{ORPC}$ – Provider

An AR is a named directed relation between two ACCO objects specified by a tuple of ACCO names regarding a QoSGroup (i.e. constant connections, local connections, RT or ORPC; selecting the implementation of the connection):

<„ProviderPDev!ProviderLDev“, „ConsumerPDev!ConsumerLDev“, QoSGroup>).

An AR is in the state „established“, if the provider is known to the consumer by name and connections are established to this consumer.

7.3.3.3.5.2.6.5 State machine GetConnectionData – Provider

Figure 41 shows the GetConnectionData state machine for the provider.

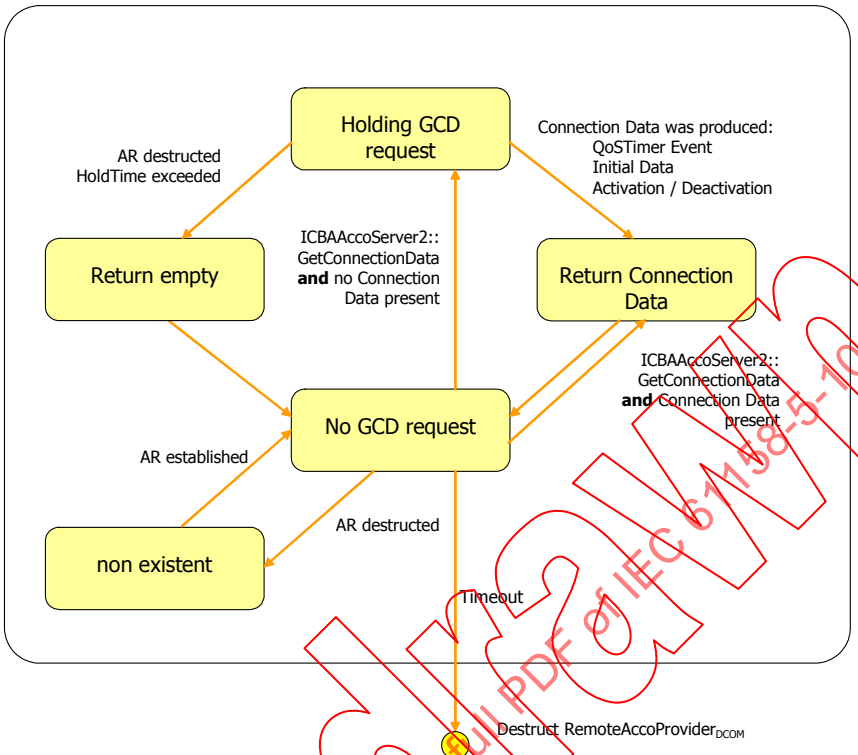


Figure 41 – State machine GetConnectionData – Provider

7.3.3.3.5.2.6.6 State machine ProviderConnection

Figure 42 shows the ProviderConnection state machine.

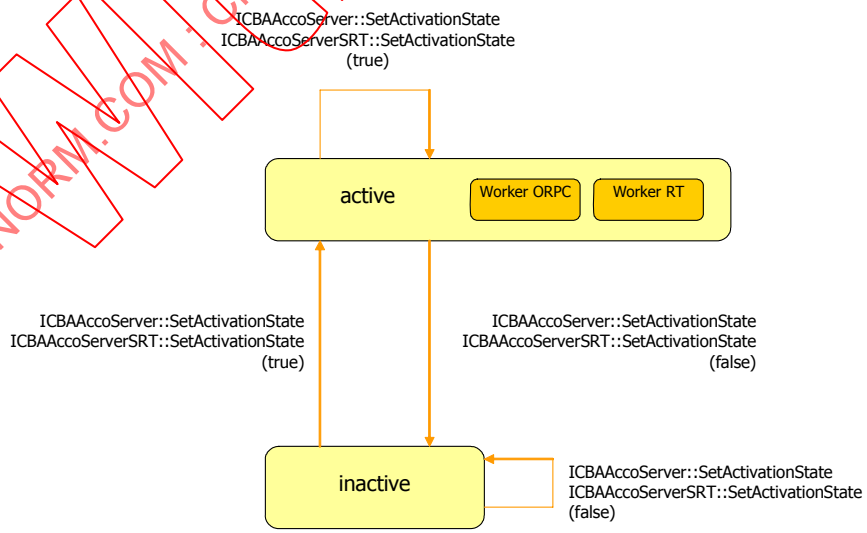


Figure 42 – State machine ProviderConnection

7.3.3.3.5.2.6.7 State machine ProvConnActivation

Figure 43 shows the ProvConnActivation state machine.

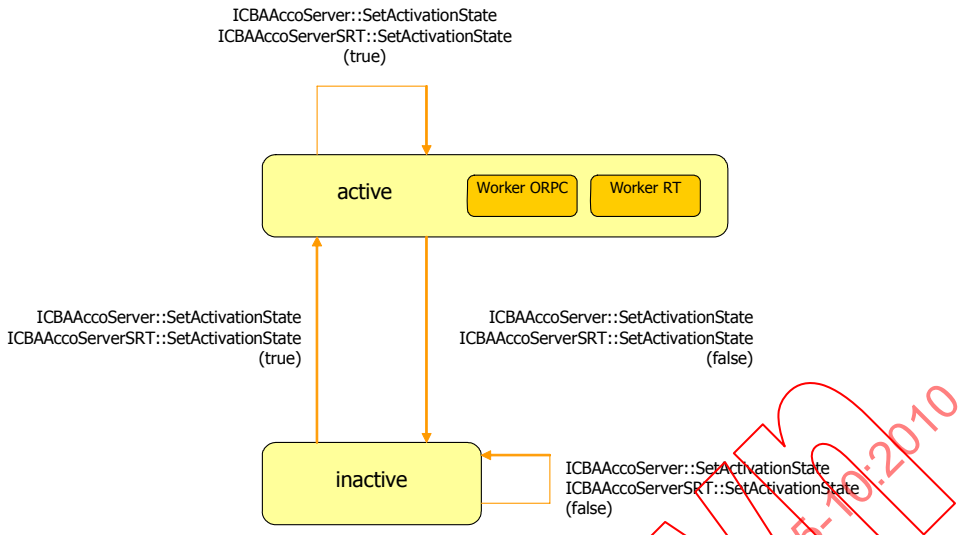


Figure 43 – State machine ProvConnActivation

7.3.3.3.5.2.6.8 State machine WorkerORPC

Figure 44 shows the WorkerORPC state machine.

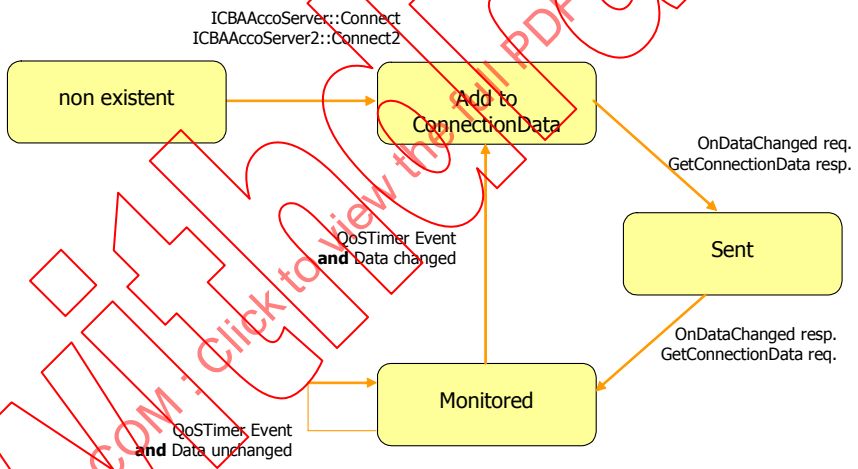


Figure 44 – State machine WorkerORPC

7.3.3.3.5.3 RT communication channel

7.3.3.3.5.3.1 General

The RT communication channel shall be based on 0 and 6.3.14.3.

Figure 45 shows the communication stack of distributed automation devices, especially for the integration of the RT communication channel.

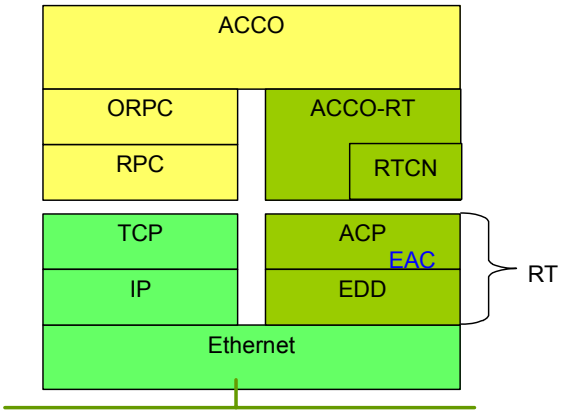


Figure 45 – Communication stack of distributed automation devices

The following stack elements have to be implemented for using the RT communication channel.

ACP
stands for Alarms (acyclic RT), Consumer (receive side of the cyclic RT) and Producer (send side of the cyclic RT).

EAC
stands for Ethernet Access Control. EAC is the definition of a port, which the ACP needs for running RT. Each device shall comply with this port by means of inward device integration with the Ethernet device driver. Depending how thoroughly the EAC is integrated into the Ethernet device driver, it blends into the existing Ethernet device driver (for example in an embedded device), but can also be a package in its own right over and above the actual Ethernet device driver (for instance for Windows, on the NDIS port).

The more thoroughly the EAC is integrated in the actual EDD, the better the performance is likely to be. The aim is a zero-copy port.

EDD
stands for Ethernet Device Driver.

RT
stands for Real Time and covers ACP, EAC and EDD.

ACCO-RT
stands for the feature which the distributed automation runtime source provides, and which integrates the RT communicationc into the ACCO. This also includes the context management as well as the control and execution of the copy cycles.

RTCN
RTCN handles the application triggered copy of provider items and consumer items in or out of the RT frames.

7.3.3.3.5.3.2 Application relations and communication relations

An RT application relation (AR_{SRT}) is a unidirectional relation between an ACCO provider instance and an ACCO consumer instance when at least one connection from the ACCO provider to the ACCO consumer has been configured. Connections with different QoS Values can exist within a single AR_{SRT}.

The AR_{SRT} is important for reconfiguration (see 7.3.3.3.6.2.5.2) and handling of fault scenarios (see 7.3.3.3.7.4).

The AR_{SRT} is only handled via ORPC protocol elements and has no equivalent at the RT protocol level.

Within a single AR_{SRT} there exist one or more unidirectional Communication relations (CR) from the ACCO provider to the ACCO consumer. Such CRs will be referred to as AccoDataCRs. Figure 46 shows the relation between AR_{SRT} and AccoDataCRs.

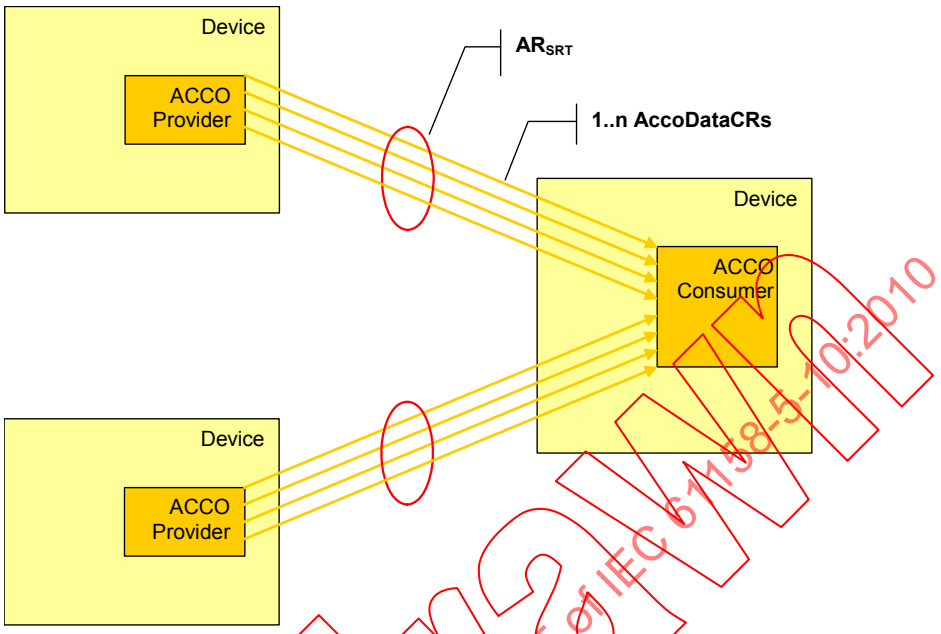


Figure 46 – Application relations between devices

The AccoDataCRs transport the connection data (value and quality code) for one or more items from the ACCO provider to the ACCO consumer in the reference data of the AccoDataCR.

The AccoDataCR is equivalent to a RT frame.

From the architectural point of view, there is no upper limit to the number of AccoDataCRs within a single AR_{SRT}. However, a device-specific upper limit will always be determined by the ACP in the light of the necessary resources.

The consumer forms the relation between the connections configured in an AR_{SRT} and the AccoDataCRs required, by reference to the following criteria:

– Each QoS Value configured in an AR_{SRT} requires at least one AccoDataCR.

– The connections configured for a given QoS Value are distributed over one ore more AccoDataCRs by reference to the size required for the necessary connection data, since there is a maximum length restriction on an AccoDataCR (see Figure 47).

  Depending on the size of the connected items, an AccoDataCR transports the connection data for one ore more connections.

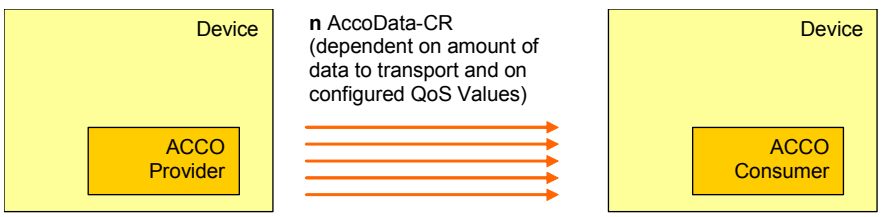


Figure 47 – Communication relations

NOTE There is no need for any user interaction or special configuration from an engineering system for distributing the connections among the AccoDataCRs.

7.3.3.3.5.3.3 Using RT

7.3.3.3.5.3.3.1 RT variants

Table 177 defines the usage of the RT variants.

Table 177 – Usage of RT Variants

RT Variant	Usage
Cyclic RT	Data connections

7.3.3.3.5.3.3.2 Frame ID

For Base Object Version greater or equal to 2.2 the Frame ID range dedicated to RT_CLASS_2 unicast is used. For Base Object Version less than 2.2 the Frame ID range dedicated to RT_CLASS_1 unicast is used. All other Frame ID ranges will not be used (see Table 366).

7.3.3.3.5.3.3.3 Segmentation

Segmentation of items over multiple AccoDataCRs is not supported out of consistency reasons.

The maximum size for the RT reference data is defined as 484 bytes.

The maximum size of an item transferred on the RT channel is defined as 450 bytes.

The minimum size for an RT frame is defined as 40 bytes. Frames with smaller reference data will be filled up to the minimum size; the fill bytes are not initialized.

Items longer than the restrictions listed above cannot be transported via the RT channel; they can only be transported via the ORPC channel.

NOTE The maximum item length can be further restricted by device-specific implementations.

7.3.3.3.5.3.3.4 RT cycle time and QoS value

In the RT communication channel the QoS is fulfilled by the following means:

– The RT cycle is shorter than the QoS Value; see Table 2-27 for details.
– RT has no explicit provider or consumer cycle; the sampling and delivery is synchronous to the application, triggered by the application itself, whereas the communication is asynchronously to the sampling or delivery.

– For an application (e.g. a PLC program), the trigger could be synchronous to the application cycle (where applicable), i.e. the cycle control point, or else by explicit use of a system call.

– For a proxy to a field device, the trigger could be synchronous to the field bus cycle.

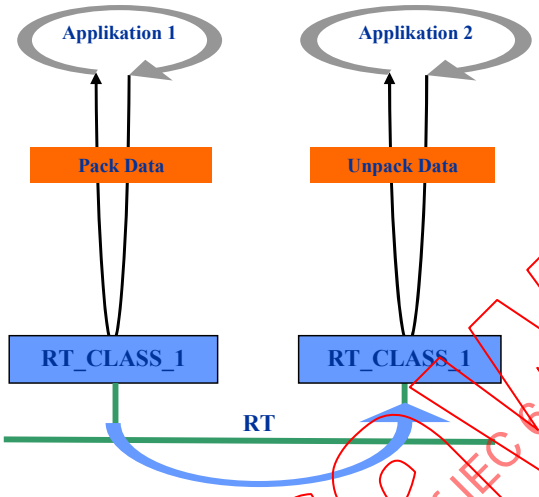


Figure 48 – RT communication channel

The RT communication channel defines the following sending clock rates for not-isochronous implementations:

1 ms, 2 ms, 4 ms, 8 ms, 16 ms, 32 ms, 64 ms, 128 ms, 256 ms, and 512 ms.

A device which supports the RT channel shall support the basic clock rate of 1 ms to effectively support load balancing. The device may specify a manufacturer-specific or product-specific minimum sending clock rate. The device shall then support all other sending clock rates above this minimum.

Table 178 shows the mapping between a given QoS Value and the corresponding RT cycle time used.

Table 178 – Mapping QoS Value to RT cycle time

QoS Value (in ms)	RT cycle time (in ms)
1	1
2	2
5	4
10	8
20	16
50	32
100	64
200	128
500	256
1 000	512

7.3.3.3.5.3.3.5 QoS violation

Since no explicit provider or consumer cycle is involved within the RT communication channel, no QoS violation will be traced via Quality Codes as in the ORPC communication channel.

7.3.3.3.5.3.3.6 Connection monitoring

7.3.3.3.5.3.3.6.1 Connection monitoring by the consumer

The RT channel offers only connection monitoring by the consumer. This generates delta information, which means that it reports if within 4 * cycle time no new data arrived for a CR or if data starts to come in again following a station failure.

Connection monitoring therefore offers the following indications to the ACP:

– IND „Failure” in the event of a station failure;
– IND „Active” when a station comes back on line.

The IND „Failure” is used to apply substitute values to the connection sinks affected, IND „Active“ is used to apply the true values. Table 179 shows the maximum apply time of substitute values in case of communication errors.

Table 179 – Maximum RT Substitute Value Apply Time

QoS Value (in ms)	Max. Substitute Value Apply Time (in ms)
1	4
2	8
5	16
10	32
20	64
50	128
100	256
200	512
500	1 024
1 000	2 048

A system specific time needs to be added to the time stated in Table 179, describing the processing time between indication of the error and application of substitute values.

7.3.3.3.5.3.3.6.2 Connection monitoring by the provider

Connection monitoring by the provider is necessary because only the consumer receives reconfiguration information from an engineering system. If communication with the provider is not possible at the time of reconfiguration, that is, if the ORPC call to the services of the ACCO Server SRT interface fails with communication error, the error scenarios are designed based on the assumption that the provider recognizes also a communication failure and disconnects the AR_{SRT}. The consumer does not notify the provider again about a change of configuration, especially when it is no longer in a relation with the provider after a reconfiguration.

Connection monitoring restricts unnecessary use of bandwidth, particularly when a consumer has been removed from the automation system.

Since cyclic RT is an unacknowledged service, the provider needs some additional means to help it decide whether the consumer is still active. The RT channel therefore defines a cyclic invocation of the Gnip service of the ACCO Callback interface on the consumer in order to enable the provider to monitor AR_{SRT} availability. In the case of a negative confirmation the whole AR_{SRT} is disconnected. Since this monitoring function is not time-critical, cyclic calling with a cycle time of 10 seconds is adequate.

The cycle time shall not be slower than every 10 seconds.

NOTE A slower cycle time may entail problems as described in the following scenario. A consumer wants to delete a RT connection and has a ORPC problem while communicating with the provider. After 10 seconds it

carries out a complete connect attempt (CoCreateInstance on the PDev object, get_LogicalDevice etc.). If this attempt to connect fails, then the consumer gives up and releases its local resources, without further notifying the provider. While the ORPC connection is temporarily out, the provider could continue sending in good faith if no Gnip occurred within the 10 seconds of reconnect time.

The modeling of the monitoring enables one PDev object with multiple LDev objects, which for its part handles multiple ARs as provider with one and the same consumer, to group all ARs under one monitoring function, i.e. it can send just one Gnip on a cyclic basis for all ARs.

7.3.3.3.5.3.4 State machines

7.3.3.3.5.3.4.1 Remote interactions

Figure 49 shows the interactions beetwen the provider and the consumer.

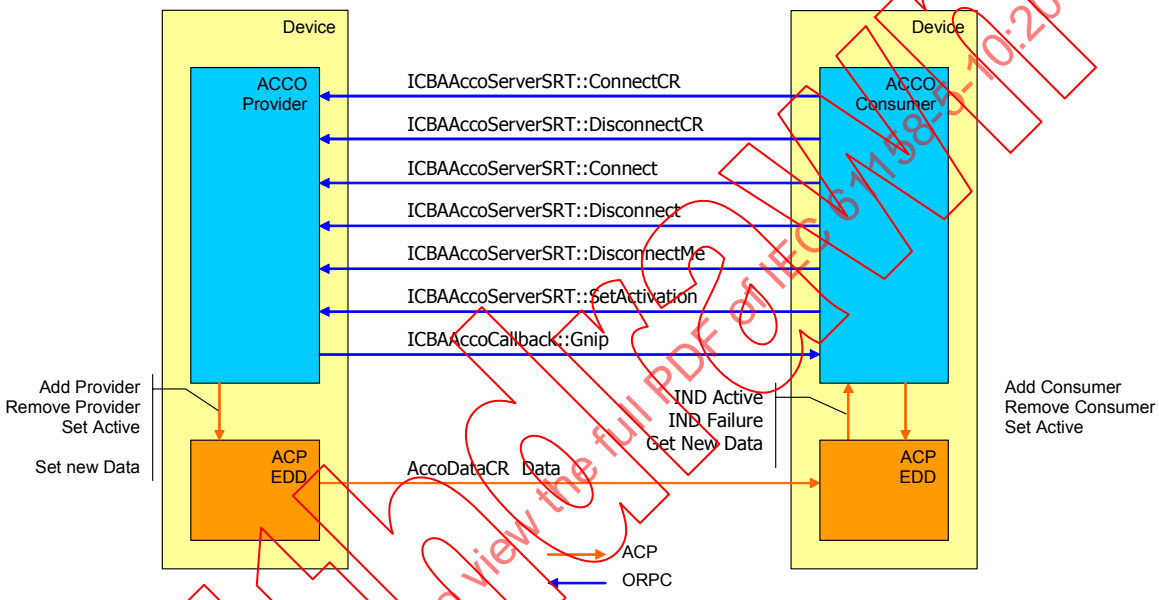


Figure 49 – Interaction between provider and consumer

7.3.3.3.5.3.4.2 State machine for application relations

The following scenarios apply to the state machine for ARs:

First contact
The consumer establishes the AR_{SRT} when an engineering system downloads connections regarding a new provider with the AddConnections service of the ACCO Management interface.

On first contacting a provider, the very next thing the consumer always does is to invoke the DisconnectMe service of the ACCO Server SRT interface in order to disconnect an AR_{SRT}, which may still exist on that provider. The provider establishes then implicitly the AR_{SRT} by means of the first ConnectCR service invocation. The establishment of the AR_{SRT} is indicated by the parameter value pFirstConnect = „true“.

Extension
The AR_{SRT} with a known provider is extended on the consumer when an engineering system downloads additional connections with the AddConnections service of the ACCO Management interface.

The AR_{SRT} is extended on the provider by means of a ConnectCR service invocation. This is indicated by the parameter value pFirstConnect = „false“.

If the provider returns pFirstConnect = „true“ in case of an extension, the consumer shall use the DisconnectMe service of the ACCO Server SRT interface to explicitly disconnect the AR_{SRT} newly established with the provider, and reestablishes all connections for the AR_{SRT}.

Reconfiguration may be necessary when there is any change to an AR_{SRT}. For further details on the subject, see 7.3.3.3.6.2.5.2.

Deletion
The AR_{SRT} with a known provider is changed on the consumer when an engineering system removes connections with the RemoveConnections service of the ACCO Management interface.

The AR_{SRT} is changed on the provider by means of the services DisconnectCR or Disconnect of the ACCO Server SRT interface for the last item in an AccoDataCR.

Reconfiguration may be necessary when there is any change to an AR_{SRT}. For further details on the subject, see 7.3.3.3.6.2.5.2.

Deleting the last connection
When the last connection in an AR_{SRT} is deleted by an engineering system, the AR_{SRT} is implicitly disconnected.

The provider implicitly disconnects the AR_{SRT} when the DisconnectCR service of the ACCO Server SRT interface for the last AccoDataCR is invoked. If a positive confirmation of this service is received, the consumer also disconnects the AR_{SRT}.

If a negative confirmation is received, the consumer waits a specific time depending on the error (see 7.3.3.3.7.4.2 for details) and then tries another connect attempt followed by a DisconnectMe service invocation, thereby disconnecting its AR_{SRT} as a result. However, if the connect attempt fails, the consumer goes ahead with disconnecting its AR_{SRT}. When the one-way communication failure from the consumer occurs, the provider detects the disconnection by means of the negative confirmation of the Gnip service.

When the AR_{SRT} is disconnected, all the objects dependent upon it (that is, the associated AccoDataCRs) are implicitly disconnected as well.

However, if new connections are established by the provider before the Gnip service of the ACCO Callback interface is called, this follows the first contact scenario and an AR_{SRT} still in existence on the provider is explicitly disconnected by invoking the DisconnectMe service of the ACCO Server SRT interface.

Station failure
The algorithm for handling consumer-side station failures is as follows:

A “Failure” indication is reported in respect to one or more AccoDataCRs. When the first IND “Failure” occurs, a 10-second timer is started. If the AccoDataCRs concerned recover within this time, that is, if an IND „Active“ is reported for all AccoDataCRs concerned, the AR_{SRT} stays intact. On the other hand if most or all of the expected IND „Active“ do not occur within this time, all the AccoDataCRs are deemed to have failed (even those for which no IND „Failure“ had yet been reported), therefore all consumer-side RT resources are released and a complete connect attempt is executed.

The expectation behind this procedure is that if a station failure is involved, all AccoDataCRs will fail within this time, but if it is a question of a transient network problem (e.g. a bad connector contact or a switch overload) or even a temporary overload in the provider, it will recover.

This algorithm avoids additional load peaks involved in reestablishing connections every time there is a supposed station failure. If a genuine station failure is involved the attempt to reestablish the AR_{SRT} is delayed by 10 seconds, but this shall be no great loss in the event of a genuine station failure.

Since ACP reports no IND „Failure“ in respect of a newly established consumer CR, if the provider is never sending data, the procedure described above is also used after the ACP consumer is set up, that is, following setup the AccoDataCR is first deemed to have failed and the 10-second timer is started. After expiration of the timer an IND „Active“ shall have come in, otherwise the whole AR_{SRT} is disconnected and re-established.

In all cases the error is forwarded at the moment it arises, by applying substitute values to the connection sinks as soon as the AccoDataCR fails (see 7.3.3.3.5.3.4.3), as soon as the IND „Failure“ occurs.

The provider disconnects the AR_{SRT} when it discovers a station failure in the consumer on receiving a negative confirmation after invoking the Gnip service of the ACCO Callback interface. When the AR_{SRT} is disconnected, all the objects dependent upon it are implicitly disconnected as well (as in the case of explicit AR_{SRT} disconnection by invoking the DisconnectMe service of the ACCO Server SRT interface).

In the event of a subsequent ConnectCR service invocation, if the provider returns the parameter value pFirstConnect = „true“ when the consumer expects „false“, the provider has previously detected the disconnection for itself and has disconnected the AR_{SRT} referenced in the service parameters. The consumer shall then carry out a full reconnect; that is it shall reestablish all AccoDataCRs.

Figure 50 shows the state machine of the consumer.

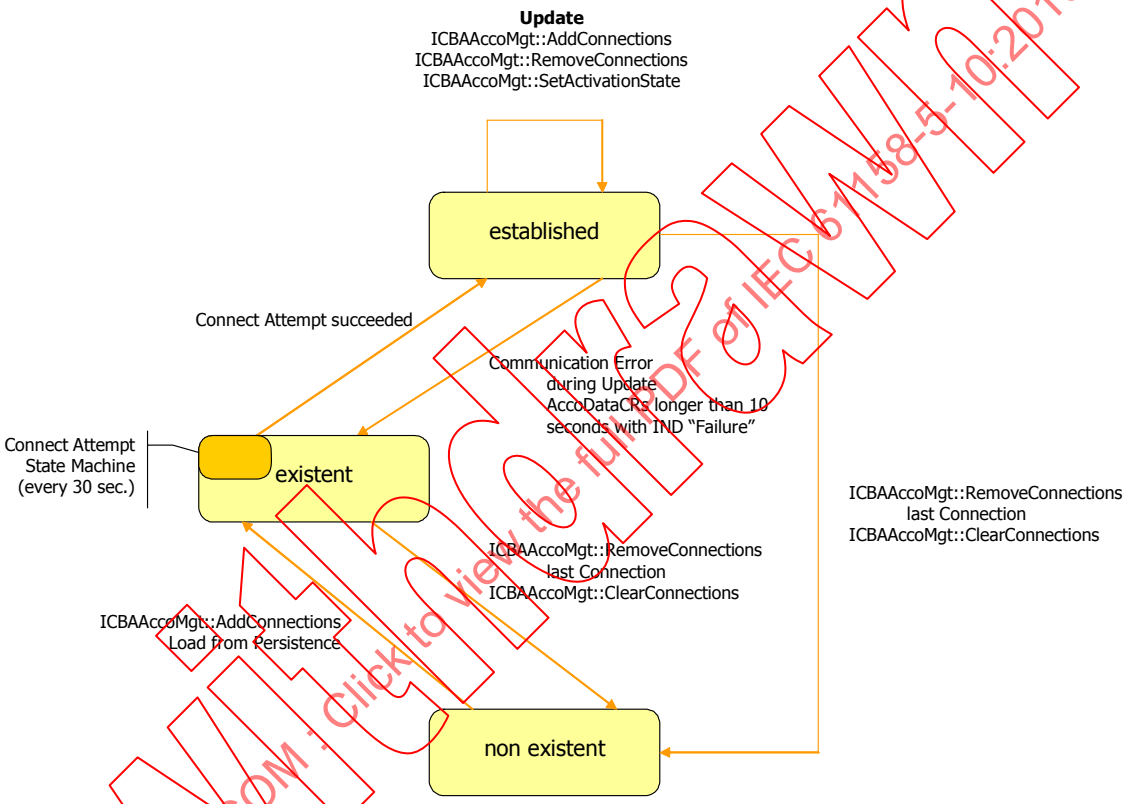


Figure 50 – State machine AR_{SRT} – Consumer

Figure 51 shows the state machine of the provider.

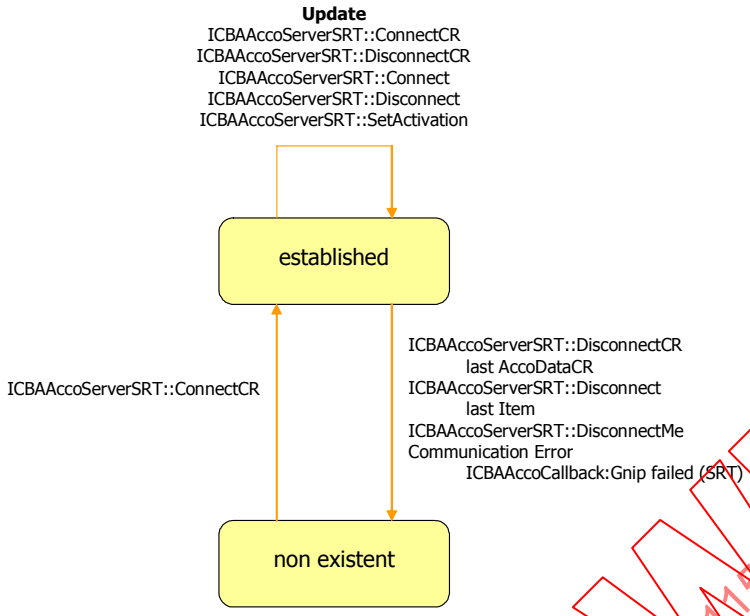


Figure 51 – State machine AR_{SRT} – Provider

7.3.3.3.5.3.4.3 State machine for AccoDataCR

The following scenarios apply to the state machine for AccoDataCRs:

Establishing, deleting and reconfiguring
The consumer sorts the items received in the course of a configuration job (service AddConnections of the ACCO Management interface), a reconfiguration or from persistence, according to their configured QoS Value and the extent of their connection data in AccoDataCRs (for information on this subject see 7.3.3.3.6.2). Reconfiguration establishes additional AccoDataCRs or deletes AccoDataCRs.

In the provider, an AccoDataCR is always explicitly established by invoking the ConnectCR service of the ACCO Server SRT interface. The AccoDataCR is explicitly deleted by invoking the DisconnectCR service of the ACCO Server SRT interface or implicitly by deleting the last items from an AccoDataCR with the Disconnect service of the ACCO Server SRT interface.

Station failure and recovery
On the consumer, all ACP indications (Failure and Active) on the AR_{SRT} for station failure and recovery are reported in an AccoDataCR. The AR_{SRT} follows the procedure described in 7.3.3.3.5.3.4.2, scenario “Station failure”, for delayed disconnection of the AR_{SRT} following a suspected station failure.

The consumer reacts to the AccoDataCR according to the indications as follows:

IND „Failure“
Substitute values are applied for all items in the AccoDataCR. The AccoDataCR is taken from the copy cycle of the consumer.

IND „Active“
The AccoDataCR is returned to the copy cycle of the consumer. Optionally when the indication is received, the new values could be sent to the items once outside the copy cycle.

The provider disconnects AccoDataCRs only implicitly on detecting the station failure in the AR_{SRT}. The AR_{SRT} follows the procedure described in 7.3.3.3.5.3.4.2, scenario “Station failure”.

Figure 52 shows the state machine of the consumer.

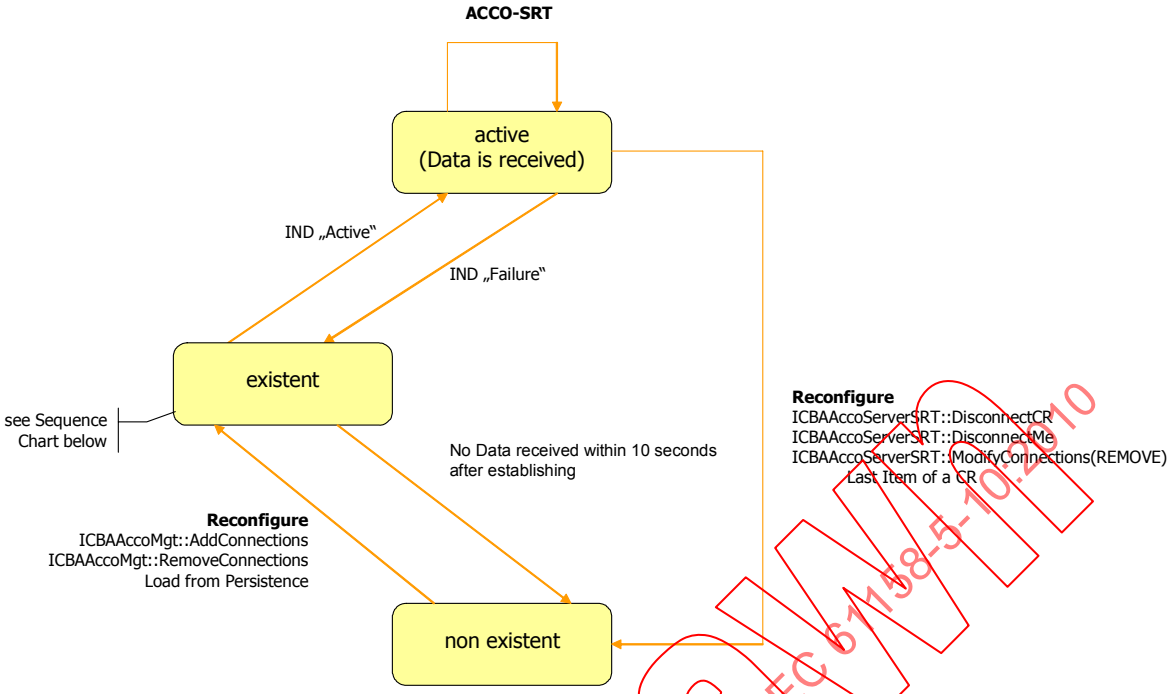


Figure 52 – State machine AccoDataCR – Consumer

Figure 53 shows the state machine of the provider.

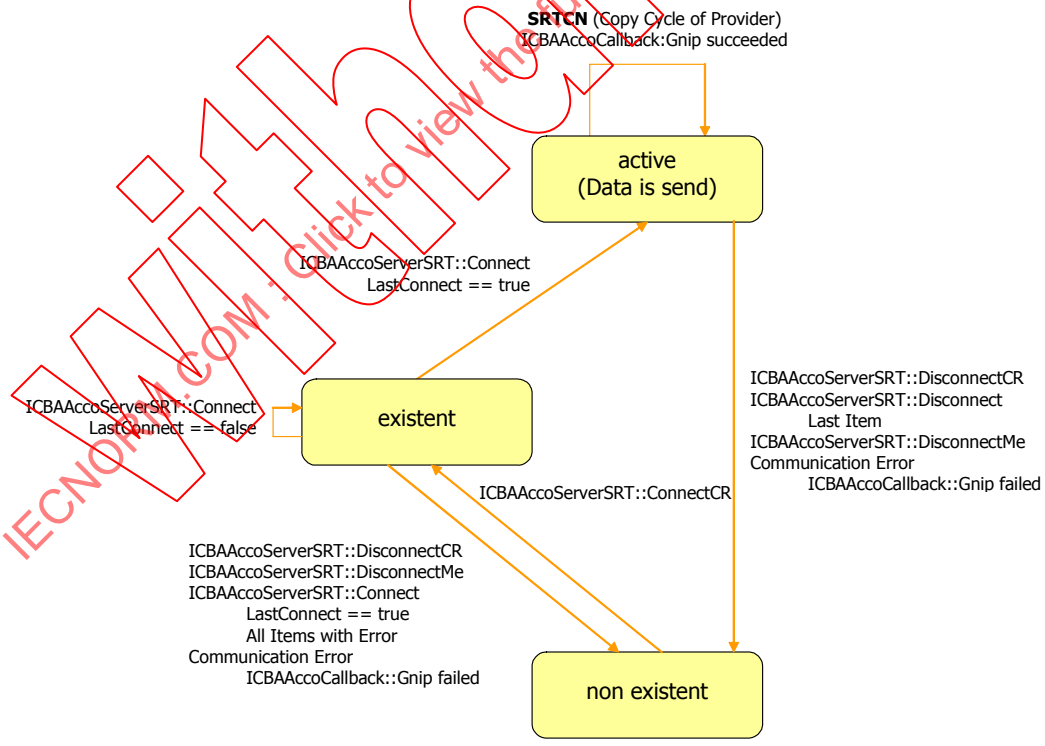


Figure 53 – State machine AccoDataCR – Provider

The consumer always establishes an AccoDataCR in the provider in the following sequence:

1. Establish the resource AccoDataCR via the ConnectCR service of the ACCO Server SRT interface.
2. Whilst there are still items to install:

- Use the Connect service of the ACCO Server SRT interface to add items that have to be transported
- Whether the LastConnect parameter is set depends upon whether another Connect service invocation follows.

The procedure can be divided into multiple Connect invocations so that an AccoDataCR can be established in manageable portions. An AccoDataCR could contain as many as 119 items of data type VT_I1 or VT_UI1 if its useful data capacity is fully utilized. Using a single service invocation to establish all the items would lead to very large PDUs.

If no items can be established for an AccoDataCR, the AccoDataCR is implicitly deleted with the Connect service (LastConnect = TRUE) of the of ACCO Server interface. This is the equivalent of the procedure when deleting the last item in an AccoDataCR via the Disconnect service of the ACCO Server interface.

After an AccoDataCR is set up by invoking the Connect service with the parameter value LastConnect = TRUE, items can only be deleted from the AccoDataCR. Its is not possible to add further items after invoking Connect with LastConnect = TRUE.

Provider and consumer reach the same understanding of the AccoDataCR's layout by the following means:

– The length of the AccoDataCR to establish is transmitted with the ConnectCR service. The length transmitted is computed by the consumer by adding up all items according to IEC/TR 61158–1:2010.
– The offset of a specific item is computed by adding up the size of the overall header plus the individual item length of all items to connect up to this specific item. The items are installed in order of Connect requests, and within a Connect request in order of the CONNECTINSRT array.
– If an item is not found on the provider, the space needed for the item will yet be reserved; computed from the transmitted extended type description. "Null Data" (all bytes including data record header) will be transmitted instead of the item; the partial result of this item notifies the consumer.

Figure 54 shows the RT frame layout.

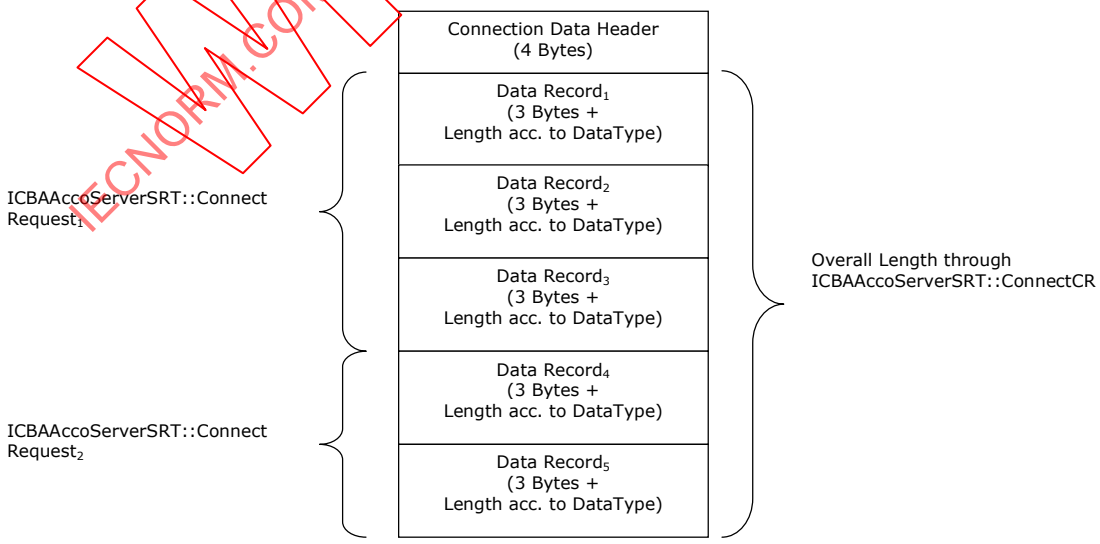


Figure 54 – RT frame layout

Figure 55 shows the communication sequence between the ACCO consumer, the ACCO provider and the respective ACP/EDD due to an AddConnections service invocation.

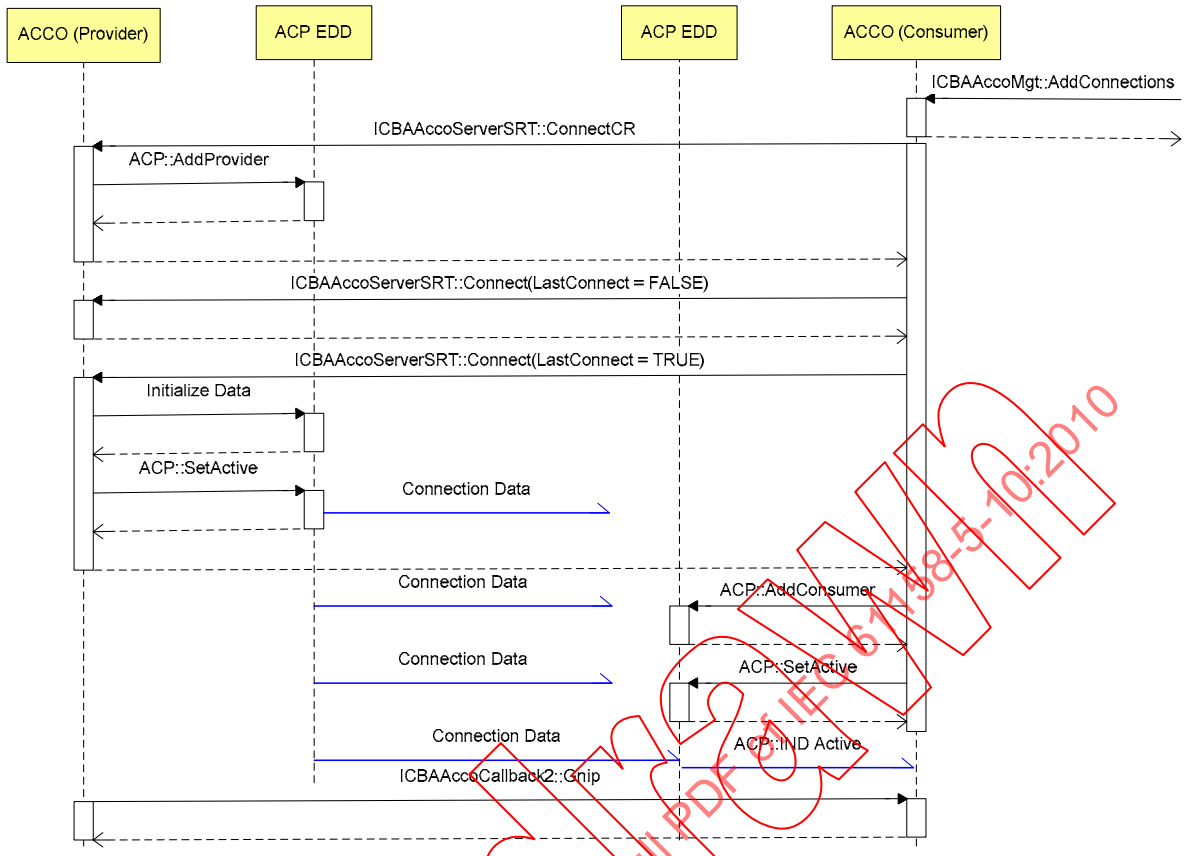


Figure 55 – Establishing an AccoDataCR

7.3.3.3.5.4 Local communication channel

7.3.3.3.5.4.1 Overview

For ORPC connections local ORPC is used. Similarly, in the case of RT it is necessary to provide a local RT. However, this functionality shall be implemented by devices, which can operate multiple applications in parallel.

Figure 56 shows a flowchart of the main copy cycle for local connections:

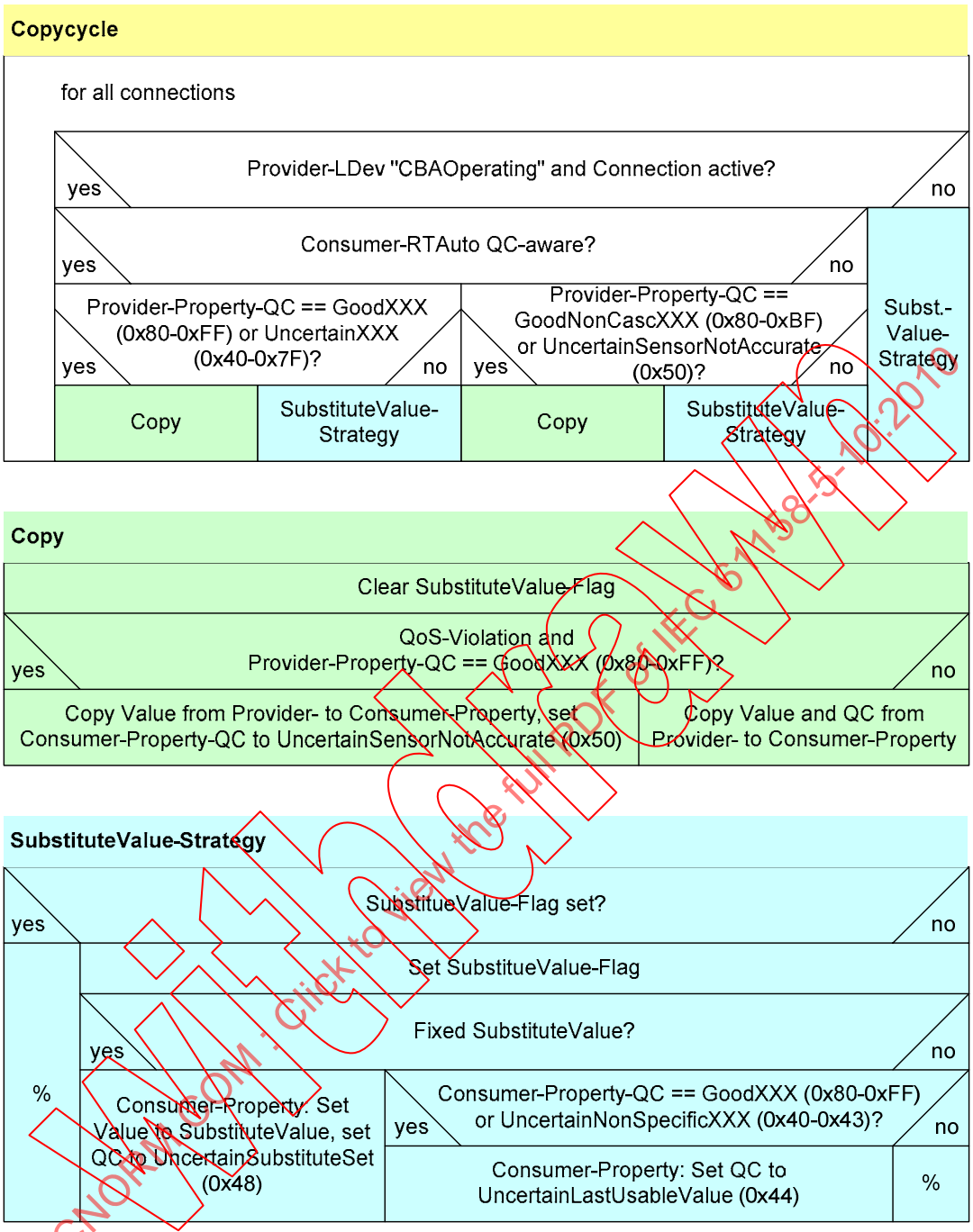


Figure 56 – Flowchart of the copy cycle for local connections

7.3.3.3.5.4.2 QoS

The local communication channel has no explicit provider or consumer cycle; the sampling and delivery shall be synchronous to the application, triggered by the application itself, whereas the communication is asynchronously to the sampling or delivery. This model is the same as in the RT communication channel.

NOTE The distributed automation runtime source should use the same trigger for both channels. For an application (e.g. a PLC program), the trigger could be synchronous to the application cycle (where applicable), i.e. the cycle control point, or else by explicit use of a system call. For a proxy to a field device, the trigger could be synchronous to the field bus cycle (e.g. at the end of the cycle) or else by using an unsynchronized copy cycle between field bus data and RT frames. The quality of the transfered data is influenced by the used implementation scheme and has to be added by the user to get to complete latency.

7.3.3.3.5.4.3 QoS violation

Since no explicit provider or consumer cycle is involved within the local communication channel, no QoS violation will be traced via Quality Codes as in the ORPC communication channel.

7.3.3.3.5.5 Constant communication channel

This value of the constant is copied once with Quality code “Good (NC) / okay” if the connection state is active resp. “Uncertain / Substitute Set” if the connection state is inactive to the connection sink in the following cases:

- Establishing the connection with a constant via the AddConnections service of the ACCO Management interface.
- Activating the connection with a constant via the SetActivationState of the ACCO Management interface
- Startup of a device with a persistent connection with a constant

Since the constant communication channel does not have any communication, no explicit QoS, detection of QoS violation, connection monitoring of provider and / or consumer is supported.

7.3.3.3.6 ACCO Management operation

7.3.3.3.6.1 Overview

7.3.3.3.6.1.1 Tasks of the consumer

The consumer implements the configuration of connection and the connection sinks. It has the following tasks:

- It shall manage the configuration information and provides for its persistence.
- It shall establishe the connections based on the configuration information with the appropriate provider.
- It shall maintain the connection sinks of data connections – initiated by the reception of new connection data from a provider. It routes the incoming connection data to the corresponding methods in the interconnected RT-Auto objects.
- It shall monitor the configured provider for failures and responds accordingly.
- It shall initiate a specific reaction to a corresponding Quality code.
- It initiates a bus-specific optimization procedure and negotiates an optimized communication channel with the provider for this purpose.

7.3.3.3.6.1.2 Tasks of the provider

The provider part of the ACCO ASE provides the necessary services for connection sources. It has the following tasks:

- It actively performs the data connections according to the Quality of Service in cooperation with the operating system – for example via timers, interrupts, etc.
- It determines the quality code of a QC-unaware RT-Auto object.
- It transfers the connection data to the consumer of the connection.
- It optimizes data connections by automatic grouping.
- It responds adequately to a consumer failure.

7.3.3.3.6.2 Establishing connections

7.3.3.3.6.2.1 Overview

By means of the AddConnections service the configuration process downloads the configuration information into the individual ACCO ASEs. Each ACCO ASE only gets the section of the configuration information for which it controls the connection ends. It does not obtain the configuration information of the entire system.

The engineering system shall group its AddConnections service calls by providers and deployed QoS to avoid extensive sorting at the consumer.

The consumer initializes its internal data with the connection sinks (i.e. an identification of the method that is to be called that was resolved in an internal form). A ConsumerID that is assigned by the consumer unambiguous identifies (within the local scope of this ACCO ASE) the internal data of a configured connection sink. This can be an array index, for example. The unambiguity shall be ensured locally by the implementation.

The consumer shall deny the request for a further data connection to an already used connection sink when the connection is established with CBA_E_INUSE.

NOTE A dual connection would mean that several sources write to a destination data item and is therefore a consistency problem.

The connection sinks are initialized with value and QC regarding to the substitute value strategy (see 7.3.3.3.4.8 for details).

The consumer shall respond with an error result to an AddConnection service in case of an error during the setup of the connections. The client can synchronously see in the error results that the connection could not be set up. Consequently, resources have not be created either for this connection.

Dependent whether a connection is established on behalf of the QoS type using the ORPC, RT or constant communication channel, the negotiation of the connection with its provider occurs according to the following subclauses. Since no explicit QoS type is given for establishing a local connection, the decision whether the local communication channel is used is made upon the value of the "Provider".

The following negotiation occurs asynchronously to the invocation of the AddConnections service.

7.3.3.3.6.2.2 Connections with a constant

Connections with constants are also established via the AddConnections service of the ACCO Management interface, such as connections with a remote or local partner.

The constant is transmitted via the service parameter SubstituteValue. The parameters to be set specifically are marked bold as follows:

```
ICBAAccoMgt::AddConnections(
  Provider          "CONSTANT!!"  // always fixed string
  QoSType             0x20          // Connection with a constant
  QoSValue            0x0
  State                             // as defined
  Count                             // Number of following structs (as defined)
  For each ADDCONNECTIONIN
    ProviderItem      ""            // always empty string
    ConsumerItem                    // Contains connection sink (as defined)
    Persistence                     // as defined
    SubstitueValue                  // holds constant with type according to
                                    // type of ConsumerItem
    EpsilonValue                    // always empty
  For each ADDCONNECTIONOUT
```

```
    ConsumerID                  // as defined
    Version                     // as defined
    ErrorState                  // as defined
);
```

The string “CONSTANT!!” shall be supplied for the parameter Provider.

NOTE The string is – as any other identifier – tested case-insensitive, but stored case-preserving.

The QoS type “0x20” indicates a connection with a constant (see 7.3.3.3.4.4). Only QoS Value "0x0" is allowed within that QoS type, otherwise CBA_E_QOSVALUEUNSUPPORTED shall be returned.

The supplied string for the ProviderItem shall be checked only syntactically for being an empty string. If some other string is supplied, CBA_E_MALFORMED shall be returned.

ConsumerItem is used as for all other connections.

Persistence is used as for all other connections. This implies that a connection with a constant reacts in the same way to the Save service of the Persist interface as all other connections, thus being also persistable.

The constant value for the connection is supplied via the substitute value as a VARIANT with its type equal to the type of the consumer item (just as a substitute value for any other connection). If the type is not equal, CBA_E_INVALIDSUBSTITUTE shall be returned. The connection with a constant reacts in the same way on scenarios, where a substitute value gets assigned, as all other connections – where applicable.

Only an empty epsilon value (a VARIANT with type VT_EMPTY) is accepted for a constant connection, otherwise the error CBA_E_INVALIDEPSILON shall be returned.

7.3.3.3.6.2.3 Negotiating with the provider – common definitions

7.3.3.3.6.2.3.1 General

The following subclauses apply to all remote communication channels (ORPC and RT).

7.3.3.3.6.2.3.2 Connect attempt

The connect attempt describes the sequence of service invocations by a consumer to connect with a provider.

Figure 57 shows the Connect Attempt State machine.

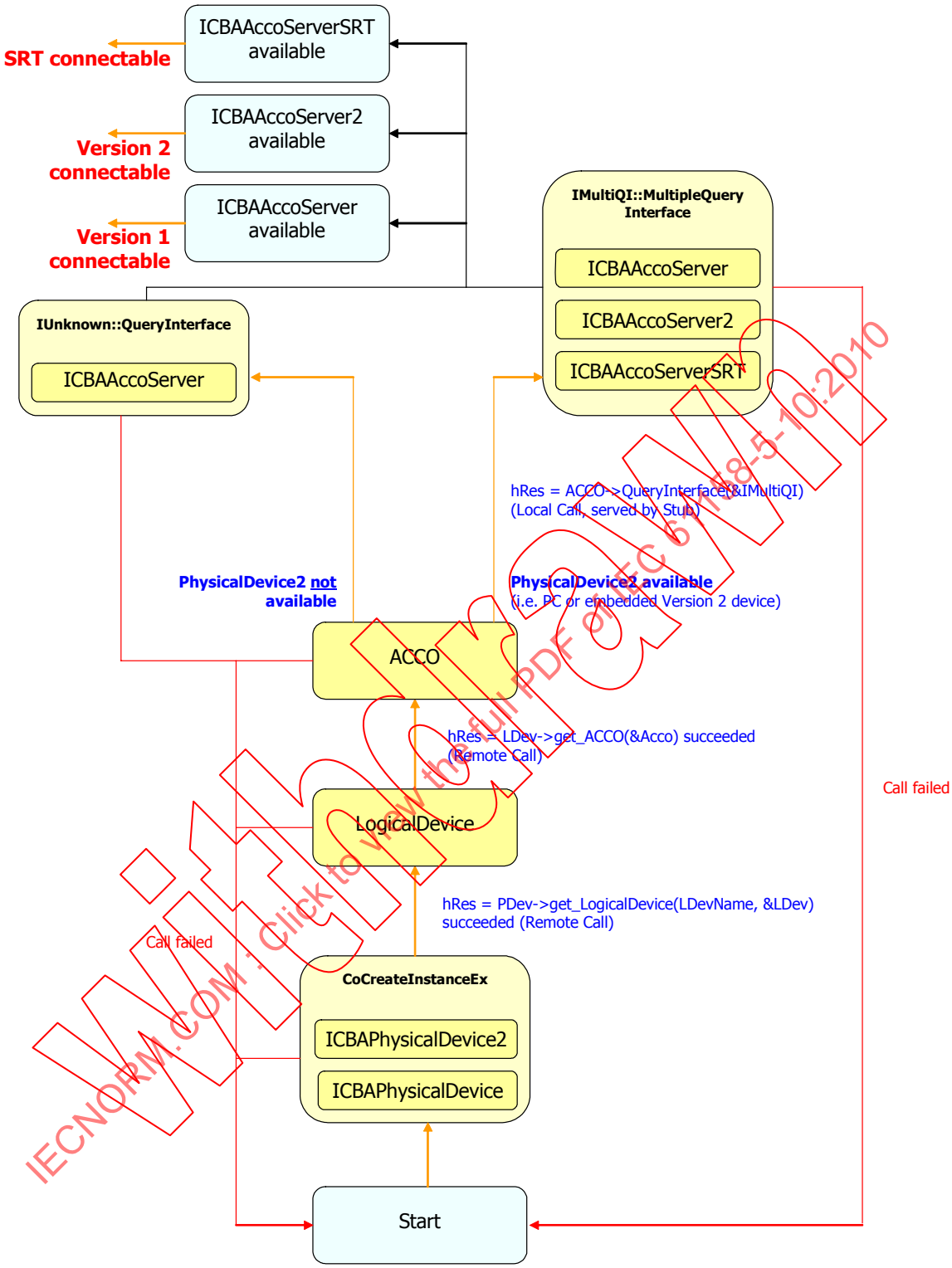


Figure 57 – State machine connect attempt

If an error occurs at any point during the connect attempt, the connect attempt shall be aborted and the error shall be noted at all affected connections. Errors, which are detected at the HRESULT parameter of a single service confirmation are noted at all attempted connections. This is resulting into a GroupError reported at the ACCO object.

Irrespective of the error that was returned at the connect attempt, a new connect attempt shall be started after a certain time has elapsed, trying to correct the situation. This shall be done even if the object or a specific property could not be found or if for example one or both

interfaces ICBAAccoServer2 or ICBAAccoServerSRT are not supported by a specific device. Therefore it is possible to update the software, firmware of the device or even to exchange the device without having to download the connections again. An error shall be returned at GroupError until the fault(s) has (have) been eliminated. In addition, an event is produced upon each change. All faults of type RPC_* can be considered as one fault.

A newly loaded connection will carry the state CBA_S_ESTABLISHING until the first connect attempt succeeded or failed. After the first connect attempt the connection carries the returned state from the connect attempt (S_OK if succeeded, or the error from the connect attempt).

The time that elapses until a failed connect attempt is retried depends on the specific error (see 7.3.3.3.7.4.2 for details).

If the provider is currently not available for communication (shown by a negative service confirmation with HRESUL value RPC_*), the further procedure follows the error scenarios described in 7.3.3.3.7.4.

7.3.3.3.6.2.3.3 Signature identity check

7.3.3.3.6.2.3.3.1 Overview

The provider has to check for signature identity by means of evaluating the data types when a data connection is established by the consumer.

Since the services Connect and Connect2 of the ACCO Server interface differ in their ability for transferring the data type, the differences are shown in the following subclauses.

7.3.3.3.6.2.3.3.2 Signature identity check for data connections (Base object version 1)

Since a VARTYPE is transferred with the Connect service, a signature identity for connections can at least be obtained with the data types VT_BOOL, VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8, VT_DATE and VT_BSTR.

If there is no signature identity, the provider shall reject the connection and a corresponding error (type mismatch) is reported. A substitute value was already applied to the consumer at the start of the connect attempt.

The following special treatments are necessary for the signature identity of arrays, strings and structures:

– Due to its structure, the SAFEARRY data type has more freedom. It is neither restricted in the dimension nor in the data type or the number of elements of the individual dimensions. Its definition does not make it particularly suitable for automation applications (e.g. in a PLC programmed to IEC 61131-3) that typically assume an array with defined dimension, data type and number of elements per dimension.

  Since the Automation data type VT_SAFEARRAY does not explicitly contain the base data type - this is a feature of the SAFEARRAY (as structure element) during the transfer - it is defined that the "VT_ARRAY | base data type" flag pair is transmitted for adjusting the type information in order to express the consumers expectation of a fixed layout. Any errors due to the dimension or the number of the elements of a dimension can only be detected during the transfer of the data, and will then be rejected by the consumer with the corresponding error message (type mismatch).

– Strings (VT_BSTR) are limited to a maximum length. Since the maximum length cannot be transmitted with the Connect service of the ACCO Server interface, any errors due to passing the maximum length of the connection sink can only be detected during the transfer of the data, and will then be rejected by the consumer with the corresponding error message (type mismatch).

– Structures (VT_USERDEFINED) are not supported in devices with Base Object Version less than 2.

7.3.3.3.6.2.3.3.3 Signature identity check for data connections (Base object version 2)

The service Connect2 of the ACCO Server interface and the service Connect of the ACCO Server SRT interface transfer the extended type description of each item as defined in 7.2.3. The expected type description has to be checked against the type description of the provider's property.

If a type includes no BSTR, a simple memory compare can be used to ensure type safety for both simple and complex types. If a BSTR is contained in the provider's type description, the type description has to be checked one by one. For BSTRs the consumer's maximum length has to be greater or equal to the provider's maximum length, since it is possible to connect a "shorter" string by its maximum length to a "longer" string.

The opposite is not allowed, since it would depend on the length of the transmitted string, whether a substitute value or the transmitted value is applied to the consumer's property.

If there is no signature identity, the provider rejects the connection and a corresponding error (type mismatch) is reported. The consumer then creates the substitute value.

7.3.3.3.6.2.4 Negotiating with the provider – ORPC

The consumer negotiates the configured connections with the appropriate providers indicated through the provider's identifier. It invokes the Connect2 service of the ACCO Server interface on the provider. The connection sources, QoS type, QoS Value and the ConsumerIDs that were set up beforehand are transferred as parameters. Errors that manifest in the course of this connect phase can be determined via the GetIDs service of the ACCO Management interface. The individual errors shall be stored at the individual connections.

NOTE If the provider is a device with Base Object Version less than 2, the consumer should invoke the Connect service of the ACCO Server interface to negotiate the connections with the provider.

The provider checks whether the given connection sources exist and store the necessary information in its internal data structures.

On the basis of its internal information (e.g. the interface description information delivered by a typelibrary), the provider recognizes via the custom interfaces of the RT-Auto objects that are to be interconnected whether the connection source is an event or a data item.

For data sources, the provider shall determine the sending of data time triggered according to the Quality of Service parameter.

With the response of the Connect service the provider delivers for each requested connection source a HRESULT and a ProviderID. The ProviderID unambiguously identifies the internal data of the provider. It is used for identifying the connection sources at all other services of the ICBAAccoServer interface.

During productive operation, the ConsumerIDs will always be transferred together with the value of the requested variable. When a consumer receives a data packet, it is able to unambiguously identify the internal data on the basis of the contained ConsumerID, and can transfer the values and quality code to the corresponding access function.

Establishing new connections via the services Connect2 or Connect of the ACCO Server interface causes a first transmission of all valid data for the active connections. This enables a consumer to be linked also in the „CBAReady" state. The consumer is provided with the current values.

7.3.3.3.6.2.5 Negotiating with the provider – RT

7.3.3.3.6.2.5.1 General

By means of the AddConnections service of the ACCO Management interface the ACCO consumer receives connections that have the QoS type „Cyclic via RT“ (see 7.3.3.3.4.4) and are loaded with a certain QoS Value.

Since the item values are sent on a continuous cyclic basis, only epsilon values of the VT_EMPTY type may be configured for RT. Otherwise the connection shall be rejected with CBA_E_INVALIDEPSILON.

Only items up to a maximum size of 450 bytes can be connected.

The configured substitute value (if specified) is applied to all connections sinks.

The consumer shall arrange the connections into frames. As many connections as possible up to a maximum of 484 bytes are assigned to one AccoDataCR. The consumer defines the position to which each item will be sent within the useful data in the AccoDataCR. Both consumer and provider need to know how much space an item occupies in the transfer buffer. The extended type definition is used for this purpose.

Each AccoDataCR uses one frame ID in the area specified by the ACP. Thereafter the frame ID uniquely identifies the AccoDataCR in the connection data stream. The frame ID is not part of the persistent information.

Next, the consumer shall establish each of these AccoDataCRs, as described in 7.3.3.3.5.3.4.3.

For all items requested via the Connect service of the ACCO Server SRT interface, the provider shall check the given data types according to 7.3.3.3.6.2.3.3.3. If the type definitions agree, the item concerned can be sent; if not, it is rejected with the appropriate error and zero bytes are transmitted in that position.

At the end of the establishment sequence, the provider shall use the consumer's MAC address and the frame ID to establish the AccoDataCR as a passive ACP provider. The reference data in the AccoDataCR has to be then initialized for the first time, i.e. the first copy cycle takes place. Then the ACP provider shall be set to active, in order to ensure that uninitialized data will not be sent. From then on, the RT has to send the current data at intervals dictated by the cycle time.

Establishment and first initialization will be carried out asynchronously to the processing of the Connect service invocation with parameter value LastConnect == TRUE.

At the end of the establishment sequence, the consumer uses the provider's MAC address to establish the AccoDataCR as an active ACP consumer.

When IND “Active” is triggered in the consumer, the data shall be taken from the AccoDataCR and applied in place of the substitute values that have to be present until that moment (see 7.3.3.3.4.8.4). This shall take place shortly after establishment of the ACP consumer. If there is no IND triggered, the procedure described in 7.3.3.3.5.3.4.2, scenario “Station failure” has to be applied.

If certain items cannot be established in the provider with the aid of the Connect service (ACCO Server SRT interface), the AccoDataCR shall be established using the remaining items. Zero bytes shall be sent in the places provided for the items, which could not be established. The individual items shall be appropriately highlighted with errors in the Connect confirmation (see 7.3.3.3.5.3.4.3 for details). Attempts shall be made to establish the missing

items in intervals depending on the returned error (see 7.3.3.3.7.4.2 for details), but this time in another new AccoDataCR.

If no item could be established, the Connect service of the ACCO Server SRT interface with parameter value LastConnect == TRUE shall be confirmed negatively with CBA_S_FRAMEEMPTY and no AccoDataCR shall be established.

The “gaps” can only implicitly corrected later by means of an appropriate reconfiguration (see 7.3.3.3.6.2.5.2).

7.3.3.3.6.2.5.2 Reconfiguration

A reconfiguration could be necessary when an AR_{SRT} already exists and changes have been made to this AR_{SRT} such as connections deleted or further ones established. Connections transferable via the RT communication channel will be restricted to allow a bump free reconfiguration of RT connections in any situation.

For devices with Base Object Version 2 an AR_{SRT} is restricted to one AccoDataCR per QoS. This restricts the maximum number of connections per QoS to a maximum of 119 connections, when using items of type BYTE. For the typical connections with an 8 byte sized value, the number of connections sums up to 43.

In an AR_{SRT} several QoS may be running in parallel; e.g. the given number of connections may be used at QoS 10 ms and another additional set of connections at 20 ms.

7.3.3.3.6.2.6 Negotiating with the provider – Local

Local Connections have no remote communication, so neither communication channel, nor any procedures defined along with the communication channel are needed. Establishing a local connection is a local matter and not described within this specification.

7.3.3.3.6.2.7 Negotiating with the provider – Constant

Connections with a constant have no implicit communication, so neither communication channel, nor any procedures defined along with the communication channel are needed. Establishing a connection with a constant is a local matter and not described within this specification.

7.3.3.3.6.3 Changing connections

A connection can be changed at any time and independently of the operating state (as long as the device is operational) of the programmable controller (i.e. it can be set up, cleared, or changed in its activation state). However, a synchronization has not been defined that permits simultaneous changes of connections concerned on several devices.

7.3.3.3.6.4 Clearing connections

7.3.3.3.6.4.1 General

The RemoveConnections service of the ACCO Management interface can be used for clearing connections. The ClearConnections service of the ACCO Management interface deletes all consumer connections. In this process, the acceptance of the configured values is deactivated immediately and the configured substitute value is applied.

If an error result is returned by the response for clearing a connection, the connection could not be cleared. This means that this connection basically did not exist.

7.3.3.3.6.4.2 Negotiating with the provider – ORPC

By means of the Disconnect service (ACCO Server interface) the corresponding connections will be cleared in the provider. Values for this connection that are sent via the communication channel shall be ignored until this connection can be cleared in the provider. The connection state is inactive. Hence, the consumer ID that was allocated to this closed connection shall not be allocated anew in the meantime.

If the provider is currently not available for communication (shown by a negative confirmation with hresult = RPC_*), the further procedure shall follow the error scenarios described in 7.3.3.3.7.4.11.

Clearing an RT-Auto object from the provider shall cause a transmission of all valid data for the affected connections (with the RT-Auto object as provider) with the appropriate Quality Codes (see 7.3.3.3.4.8.10).

7.3.3.3.6.4.3 Negotiating with the provider – RT

If connections are deleted using the RemoveConnections service of the ACCO Management interface, the deletion can potentially be synchronized with the provider in one of two ways:

– By reconfiguring (establishing a set of “cleaned” AccoDataCRs without deleted connections; see 7.3.3.3.6.2.5.2), depending on the reconfiguration strategy.

– By using the Disconnect service of the ACCO Server SRT interface. In this case the provider shall send “Null Data” (analoguos to Null Data on errors during establishing an AccoDataCR; see 7.3.3.3.5.3.4.3 for details) in place of the connection data previously returned for the items.

7.3.3.3.6.4.4 Negotiating with the provider – Local

Local connections have no remote communication, so neither communication channel, nor any procedures defined along with the communication channel are needed. Clearing local connections is a local matter and not described within this specification.

7.3.3.3.6.4.5 Negotiating with the provider – Constant

Connections with a constant have no implicit communication, so neither communication channel, nor any procedures defined along with the communication channel are needed. Clearing connections with a constant is a local matter and not described within this specification.

7.3.3.3.6.5 Activating and deactivating connections

7.3.3.3.6.5.1 General

Each connection carries a status that defines whether it is active or inactive. The initial status of the connection is transferred via the AddConnections service (ACCO Management interface) when the connection is established. The status can be changed using the SetActivationState service (ACCO Management interface).

Although an inactive connection has already been established in the resources of consumer and provider, connection data is not transferred. With respect to the data, it has the same effect as if it were not established.

If a connection is inactive, the acceptance of the configured values is deactivated synchronously to the invocation of the method in the consumer, and the configured substitute value is applied.

A connection with a constant shall react in the same way to the SetActivationState service (ACCO Management interface), as any other connection. If the state gets activated, the constant shall be assigned once to the consumer item, if the connection is deactivated, the consumer item shall be set to the substitute value of the connection (this is the constant itself, see below) and the quality code shall be set to “Uncertain / Substitute Set”.

7.3.3.3.6.5.2 Negotiating with the provider – ORPC

The connection status is transferred to the provider by means of the SetActivation service (ACCO Server interface).

If the provider is currently not available for communication (shown by a negative confirmation with hresult = RPC_*), the further procedure shall follow the error scenarios described in 7.3.3.3.7.4.11.

Changing the connection state by the SetActivation servive (ACCO Server interface) shall cause a transmission of all valid data for the changed connections with the appropriate Quality Codes (see 7.3.3.3.4.8.10).

7.3.3.3.6.5.3 Negotiating with the provider – RT

The provider always shall send valid connection data, regardless of whether a connection is activated or deactivated. This applies even if an entire AccoDataCR contains deactivated items.

The consumer shall only accept connection data for activated connections from the AccoDataCR.

NOTE From the point of view of the copy cycle on the provider, this removes the necessity for an SetActivation service in the ACCO Server SRT interface, since the Activate and Deactivate functionality in respect of the connection data is only important for the consumer. Nevertheless, it is provided so that activated or deactivated items can also be recognized on provider side diagnostics.

7.3.3.3.6.5.4 Negotiating with the provider – Local

Local Connections have no remote communication, so neither communication channel, nor any procedures defined along with the communication channel are needed. Changing the activation state of local connections is a local matter and not described within this specification.

7.3.3.3.6.5.5 Negotiating with the provider – Constant

Connections with a constant have no implicit communication, so neither communication channel, nor any procedures defined along with the communication channel are needed. Changing the activation state of connections with a constant is a local matter and not described within this specification.

7.3.3.3.7 Productive operations of connections

7.3.3.3.7.1 Overview

The consumer receives data packets via the established communication channel, unpacks the individual connection data and deploys the information to invoke the corresponding method with the transferred parameters and/or stores the corresponding property. The format of the transmitted packet is specified in the service description of the OnDataChanged service.

7.3.3.3.7.2 Productive operation for data connections

7.3.3.3.7.2.1 ORPC channel

For the data sources, the data provider possesses lists that shall be sorted by communication channel and QoS. On the basis of these lists and the connection status ("active"), it identifies the data that shall newly be determined. This data shall be read via the configured access methods of the related objects.

For the QoS "cyclically on changes", this value shall be compared with the last value in a local cache for deviations by ± ε (see 7.3.3.3.4.5). A new value shall only be transferred after a change.

The values shall be transferred via the communication channel together with the corresponding ConsumerID and quality code (see 7.3.3.3.4.8). The transmission path depends on the communication process. Figure 58 shows the control flow for the productive operation of data connections on the ORPC communication channel.

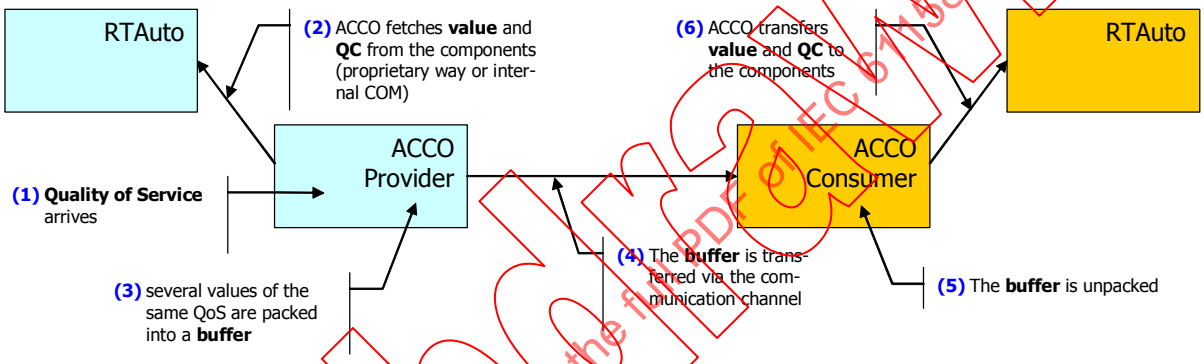


Figure 58 – Productive operation of data connections (ORPC channel)

This packet is transmitted to the consumer according to the definitions of the communication channel using the OnDataChanged service (ACCO Callback interface).

The provider part of the ACCO ASE automatically groups data connections of the same QoS and communication channel to reduce the communication loading since multiple data items are bundled and transmitted in one packet (ORPC-PDU).

The provider has to transfer the actual data of connections independent of the QoS in the following cases:

- Operating state transition from CBAReady to CBAOperating, respective CBAOperating to CBAReady with the appropriate Quality Codes (see 7.3.3.3.4.8.9). This affects all data connections.
- Establishing new connections by the Connect service (ACCO Server interface). This affects the new connections.
- Changing the connection state by the SetActivation service (ACCO Server interface) with the appropriate Quality Codes (see 7.3.3.3.4.8.7). This affects the changed connections.
- Clearing an RTAuto from the provider with the appropriate Quality Codes (see 7.3.3.3.4.8.10). This affects the connections regarding that RTAuto.

7.3.3.3.7.2.2 RT channel

Figure 59 shows the control flow for the productive operation of data connections on the RT communication channel.

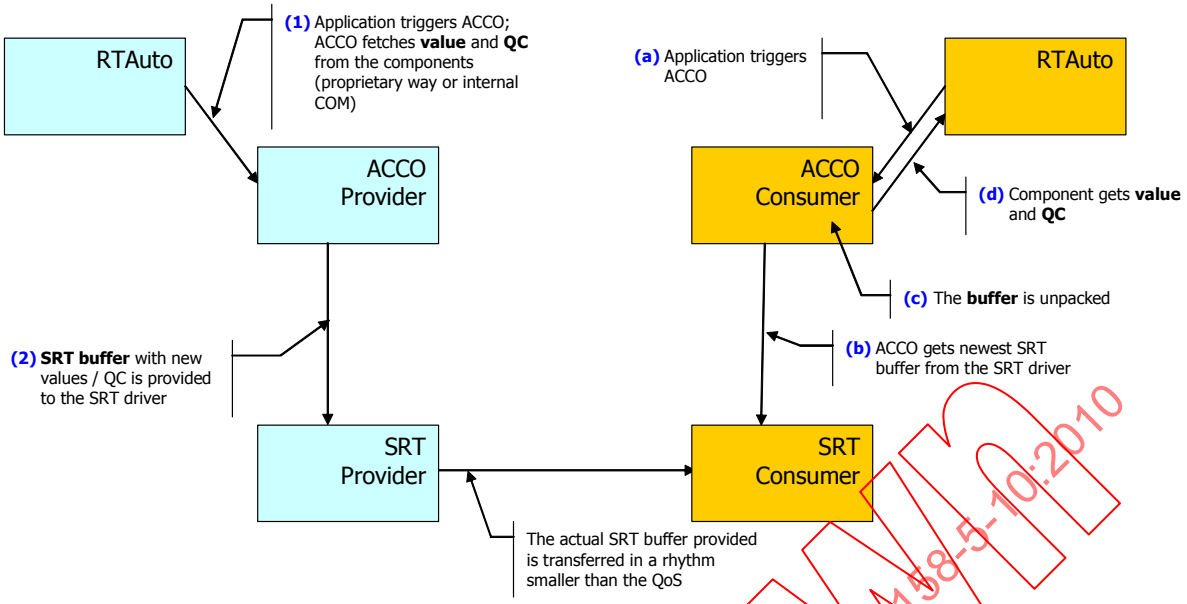


Figure 59 – Productive operation of data connections (RT channel)

In the RT channel an application trigger shall activate the connection data sampling. All RT connection data regarding this application shall be sampled into RT buffers. When the sampling is finished, the new data buffers shall be provided to the RT driver (see sequence (1) – (2) in Figure 59).

The RT buffer shall be handed over to the consumer in a rhythm derived from the quality of service (see Table 178). The RT driver shall transmit always the actual RT buffer provided.

Independent from the arrival of an RT buffer in the consumer, the application shall trigger again the exchange of data; derived from that trigger, the new data have to be unpacked from the RT buffer and delivered to the application (see sequence (a) – (d) in Figure 59).

An application trigger has also to take place in the following situations:

– Operating state transition from CBAReady to CBAOperating, resp. CBAOperating to CBAReady with the appropriate Quality Codes (see 7.3.3.3.4.8.9). This affects all data connections.
– Clearing an RTAuto from the provider with the appropriate Quality Codes (see 7.3.3.3.4.8.10). This affects the connections regarding that RTAuto.

7.3.3.3.7.2.3 Local channel

Figure 60 shows the control flow for the productive operation of data connections on the local communication channel.

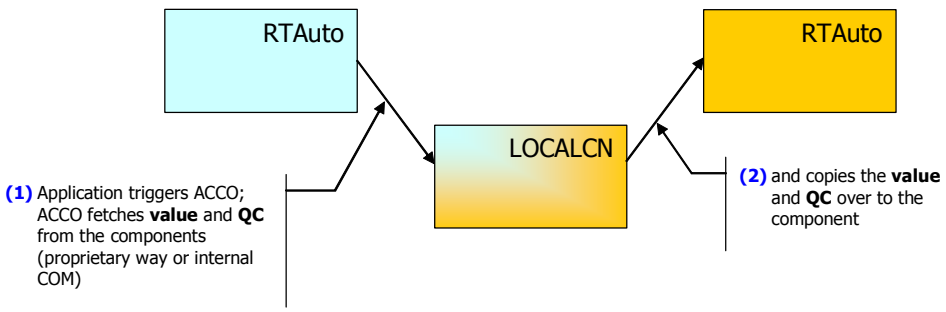


Figure 60 – Productive operation of data connections (Local channel)

An application trigger shall activate the local copy of the connection data. All local connection data shall be copied over then to the destination.

7.3.3.3.7.3 Productive operation for the RT communication channel

Figure 61 shows the data flow for the cyclic RT communication channel.

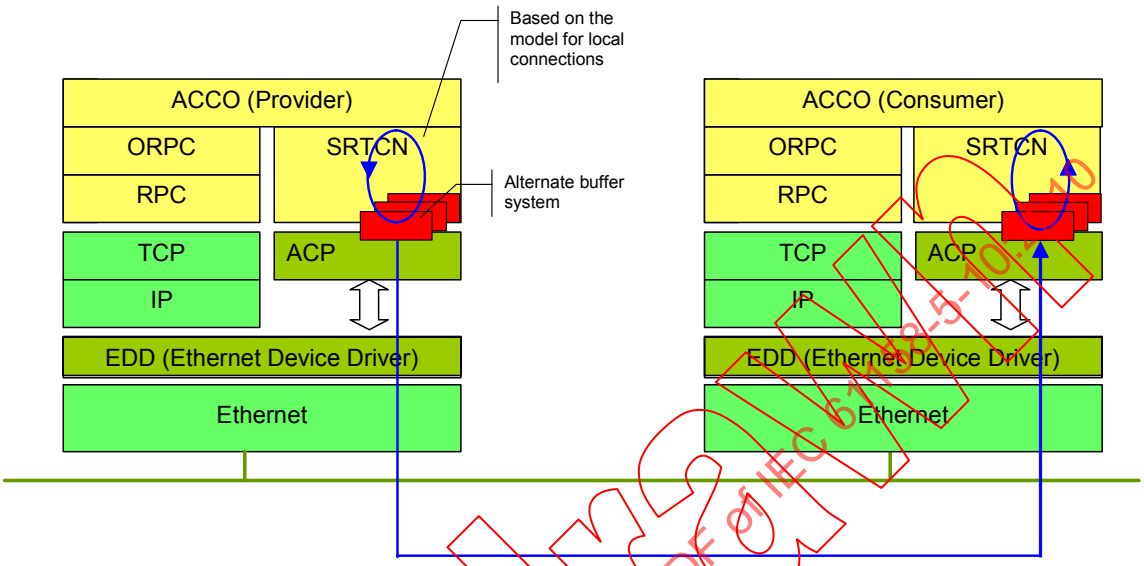


Figure 61 – Data flow for cyclic RT

The ACCO provider shall use an add provider function of the ACP to establish a cyclic job for sending certain data. This requires the parameters Send Interval, Frame ID and MAC_Address. This job can use a memory area which the RT transfers based on the send interval, whilst a second memory area can be available to the ACCO for preparing the new data (alternate buffer system). In this area, the application can copy data based on a trigger (e.g. the cycle control point of a PLC). This is the task of the ACCO-RT.

The data preparation, which the ACCO carries out, shall be arranged on rather the same model as for local connections, in which the provider side can sets up a connection from each item to the RT buffer. However, since the consumer of the connection can be any device, the connection data shall be marshaled in the local buffer. It is not enough simply to copy the values as in the case of local connections. Transfer shall always take place independently of a change of value, i.e. cyclically.

7.3.3.3.7.4 Fault scenarios

7.3.3.3.7.4.1 Overview

The fault handling is a simple error handling procedure. Any fault is handled as a device failure of the distributed automation device (power-off or defect). The fault elimination procedure does not distinguish between line problems or the failure of a device. This means that temporary line problems are handled like a failure.

7.3.3.3.7.4.2 Time intervals

Table 180 defines the time intervals and timeouts that shall be applied for devices with Base Object Version greather or equal to 2.

Table 180 – Time Intervals and Timeouts

Situation	Time Interval	Timeout
New connect attempt after communication error	30 seconds	30 seconds
New connect attempt after application error	configurable (typically 6-10 seconds)	30 seconds
Ping service (ACCO Server interface)	Ping factor * min. QoS value	Ping factor * min. QoS value
OnDataChanged service (ACCO Callback interface)	QoS value (on change)	
empty OnDataChanged service (ACCO Callback interface)	30 seconds	
Gnip service (ACCO Callback interface)	10 seconds	10 seconds
GetConnectionData service (ACCO Server interface)	QoS value (on change)	10 seconds

For devices with Base Object Version less than 2 all time intervals and timeouts shall be set to 30 seconds.

7.3.3.3.7.4.3 Packet duplication

For the ORPC communication channel, packet duplication is already handled by the IP suite ASE.

For the RT communication channel in view of the cyclic nature of the protocol, packet duplication is not a problem.

7.3.3.3.7.4.4 Packet loss

For the ORPC communication channel, packet loss and implicit retransmission is already handled by the IP suite ASE.

Since the protocol is cyclic in the case of the RT communication channel, the loss of a packet can be tolerated in certain circumstances. RT itself defines how many packets can be lost until an IND “Station Failure” is raised (see 7.3.3.3.5.3.3.6.1).

7.3.3.3.7.4.5 Garbled data

Garbled data is rejected by the Medium Access ASE and then handled in the same way as packet loss.

The ORPC communication channel provides no further safeguards against data garbling.

The RT communication channel provides no further safeguards against data garbling.

7.3.3.3.7.4.6 Sequence switching

For the ORPC communiucation channel, sequence switching is handled by the IP suite ASE. It can happen especially, if the ORPC communication channel is drawn via routers. In local networks there shall actually be no sequence switching, since switches are forced to maintain sequences in a priority class.

If packet sequences are transposed in the RT communication channel, older values for a cycle can overlay new values. To overcome the problem each RT packet transported in the RT protocol contains a sequence counter. Then at the ACP level, the sequence number prevents an older packet from overwriting a newer packet in the consumer. This property is transparent to the user, and therefore the ACCO ASE need take no action to prevent sequence switching.

7.3.3.3.7.4.7 Recycling frames

A consumer has established an AccoDataCR with the provider. The consumer then goes away due to a network outage, but the provider has not yet sent an Gnip service (ACCO Callback

interface) and so has not yet noticed that the consumer is not there. The provider therefore keeps sending, using a particular frame ID.

The situation is prevented by the first contact scenario, since the consumer sends as its first invocation a DisconnectMe service (ACCO Server SRT interface), so causing the entire AR_{SRT} to be disconnected.

7.3.3.3.7.4.8 Failure of the provider in productive operation

7.3.3.3.7.4.8.1 ORPC communication channel

The consumer detects the failure of a provider during productive operation by means of the monitoring procedure. It requested to set all data sinks that are supplied by this provider to the configured substitute value or to hold their last values (depends on the configuration). At the same time, the quality code is set to "Substitute–set" or "last usable value" (see 7.3.3.3.4.8.2).

The dead time in Figure 62 shall be set to “Ping factor * min. QoSValue” according to Table 180. Any Ping confirmation after applying substitute values is viewed as communication error, even if the Ping confirms “normally” before the communication timeout strikes; a new connect sequence is started.

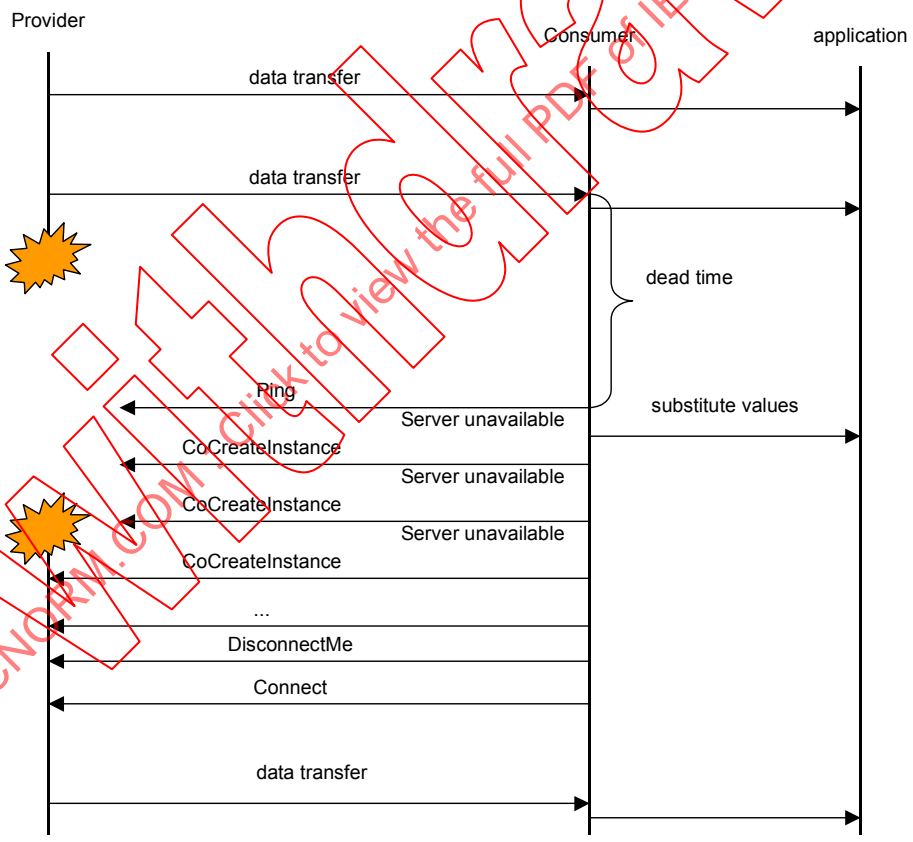


Figure 62 – Failure of the provider in productive operation (ORPC push mode)

The dead time in Figure 63 is set to 10 seconds according to Table 180.

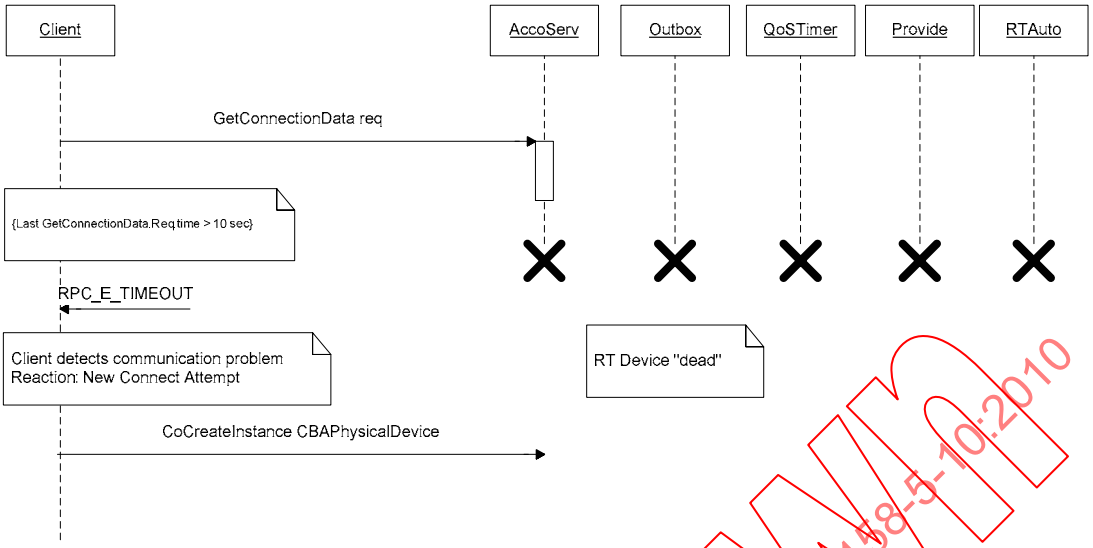


Figure 63 – Failure of the provider in productive operation (ORPC pull mode)

The consumer shall try to re-establish the connection to the provider at a certain interval, which depends on the specific error (see 7.3.3.3.7.4.2 for details).

Once the provider is available again, the consumer shall clear all connections with the DisconnectMe service and shall re-established all connections by means of the Connect service. This clears the situation in which the provider has not yet detected a communication failure.

7.3.3.3.7.4.8.2 RT communication channel

The failure of a provider in productive operation is reported to the consumer by RT via IND „Failure“. This means that no explicit ACCO monitoring is necessary on the part of the consumer (such as needed in the ORPC channel). See 7.3.3.3.5.3.3.6.1 for details.

In the event of communications failures on the consumer side involving one ore more AccoDataCRs, the procedure defined in 7.3.3.3.5.3.4.2, scenario “Station failure” has to be applied.

– If the consumer detects the loss of the provider by IND „Failure”, he shall set all connection sinks of the appropriate AccoDataCR to the configured substitute value or to hold their last values (depends on the configuration). At the same time, the quality code shall be set to “Uncertain / Substitute–set” or “Uncertain / last usable value”.

– A 10-second timer shall be started. If further data comes in within that time, as indicated by IND „Active“, this data shall be applied, the timer canceled and everything is back in order.

– If the timer expires, substitute values shall be applied to all items in this AR_{SRT}. The whole AR_{SRT} shall be then disconnected via the DisconnectMe servive (ACCO Server SRT interface) and all AccoDataCRs shall be reestablished via the ConnectCR service (ACCO Server SRT interface). If the reestablishment fails to work (including when a provider station failure is detected on the ORPC channel), a new connect attempt shall be retried after a time period depending on the specific error (see 7.3.3.3.7.4.2 for details).

– Once the provider is available again, all connections at the provider's side shall be cleared by means of the DisconnectMe service (ACCO Server SRT interface). Subsequently, all connections shall be reestablished using the ConnectCR service

(ACCO Server SRT interface). This clears the situation in which the provider has not yet detected a communication failure.

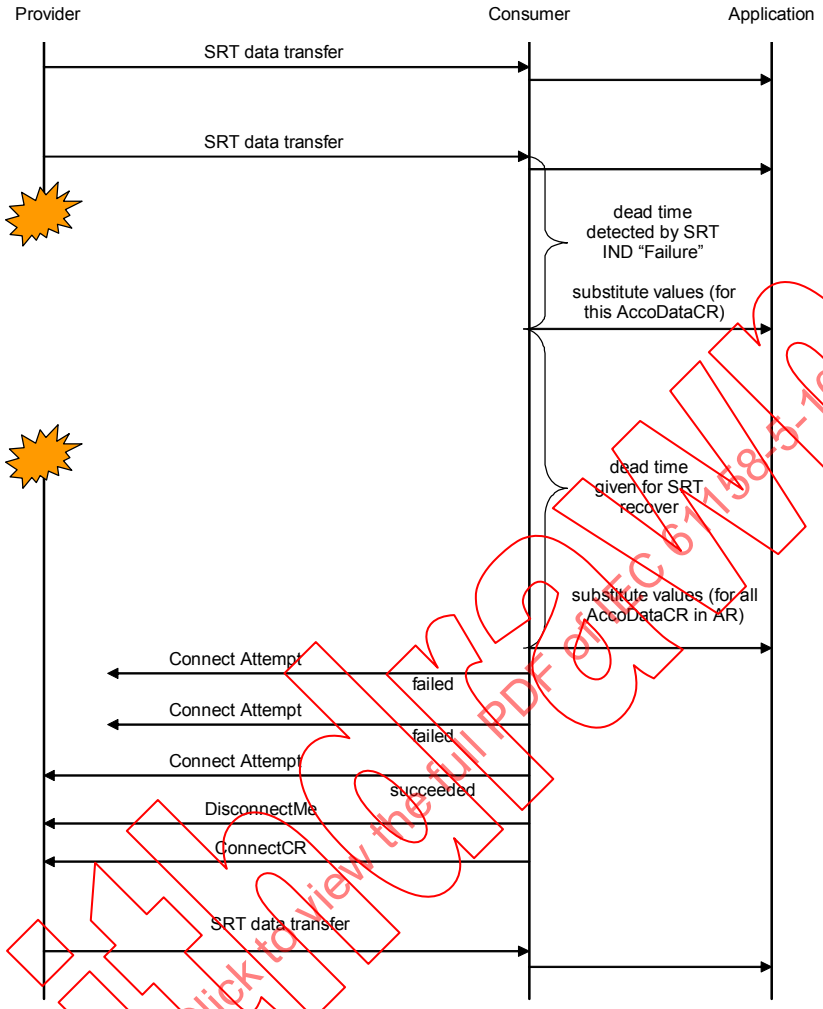


Figure 64 – Scenario 1: Provider failure in productive operation (RT)

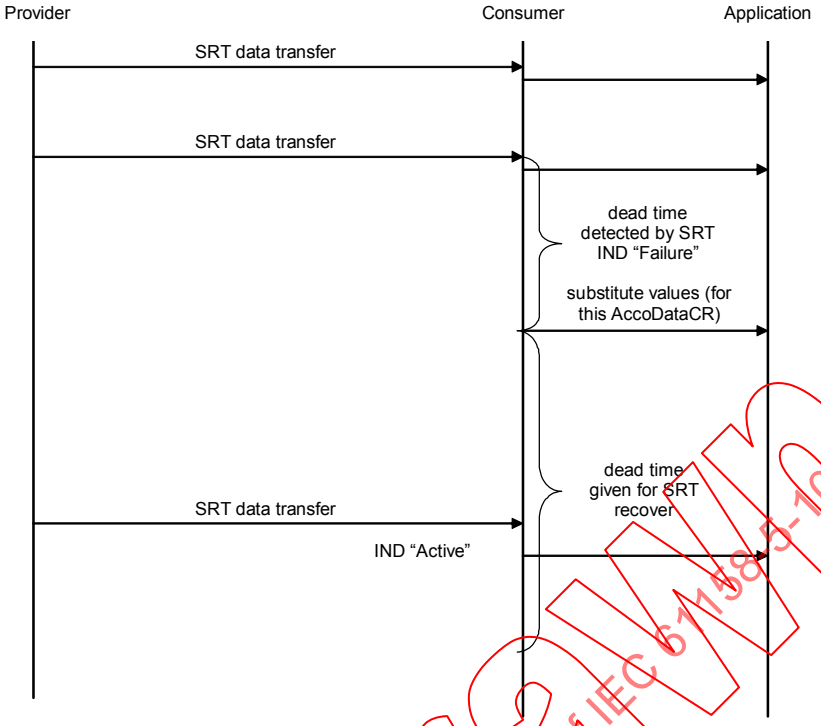


Figure 65 – Scenario 2: Recovery from provider failure in productive operation (RT)

A special situation appears when the establishment of the connection through ORPC works fine, but the communication via RT is never established. The state CBA_S_NOCONNECTIONDATA shall be applied to the connection state after the connection is established but before the first connection data is received. If no connection data is received after the specified dead time, the connection will be established anew, until the situation clears.

7.3.3.3.7.4.9 Failure of the consumer in productive operation

7.3.3.3.7.4.9.1 ORPC communication channel

If the provider detects the failure of the consumer, the provider relinquishes all the connections to this consumer. As shown in Figure 66, it deletes all connection information because the consumer is responsible for re-establishing the connections again.

Subsequently, the consumer re-establishes – as defined above – the connections.

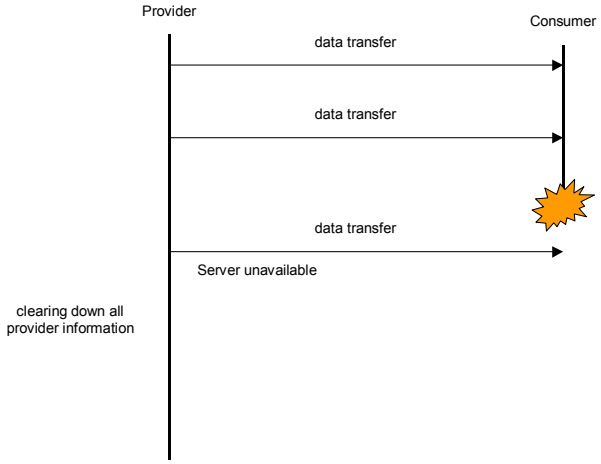


Figure 66 – Failure of the consumer (push mode)

The provider shall respond with CBA_S_NOCONNECTION in the event of an indication of a Ping service of the failed consumer (e.g. after a short failure of the link). This informs the consumer about the fact that all its connections are cleared.

In push mode when a provider sends items to a consumer changing only very slowly, the provider can not detect a failure of the consumer. Therefore the provider shall send in a specific interval (30 seconds) an empty connection data, consisting only of a header with no items. The negative OnDataChanged confirmation detects then the failure of the consumer.

Figure 67 shows the fault scenario for the pull mode.

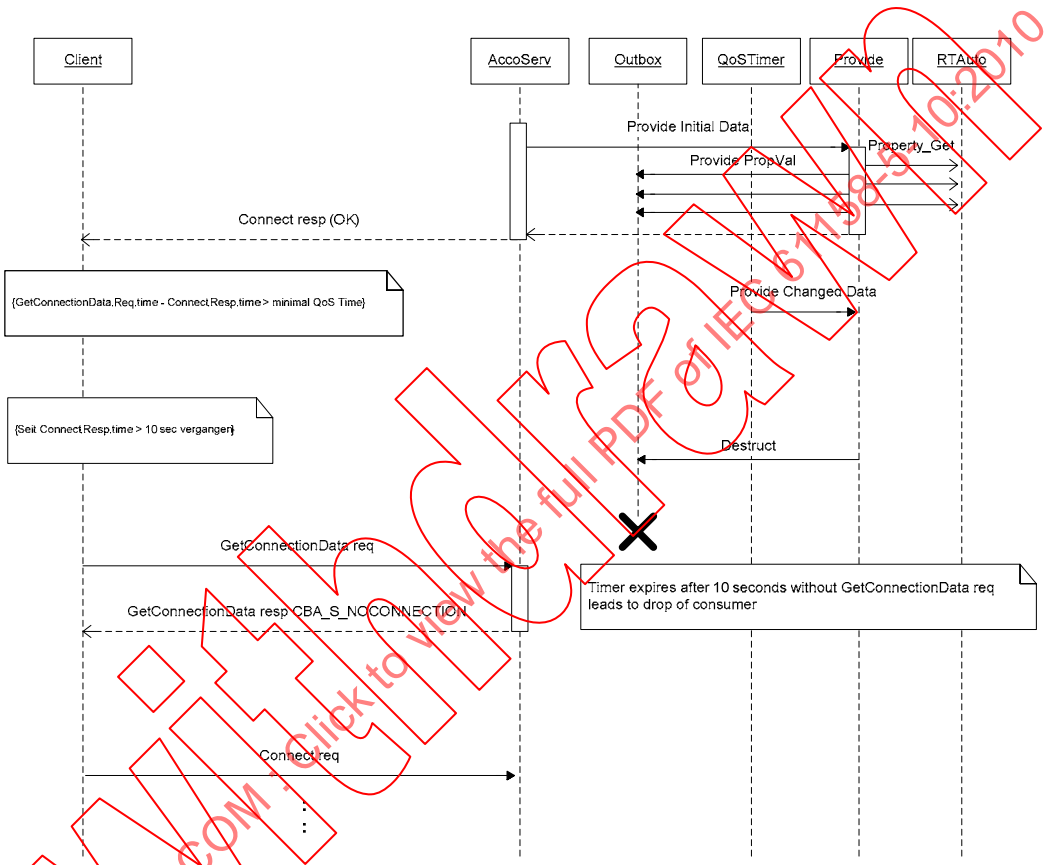


Figure 67 – Failure of the consumer (pull mode)

In pull mode the provider shall use a time with an expiration of 10 seconds wound up after responding to a GetConnectionData request (ACCO Server interface). If no further GetConnectionData request was obtained, the consumer is dropped.

7.3.3.3.7.4.9.2 RT communication channel

If the provider detects the failure of the consumer via a negative Gnip confirmation (ACCO Callback interface), it relinquishes the complete AR_{SRT}. As shown in Figure 68, it shall delete all connection information because the consumer is responsible for re-establishing the connections again.To do this the provider drops all AccoDataCRs within its AR_{SRT} by means of a remove provider function of the ACP.

Since the structure of CRs is asymmetric (one Gnip service invocation for multiple AccoDataCRs), the provider cannot selectively disconnect one AccoDataCR in particular.

Synchronization between the provider and the consumer via the AR_{SRT} relinquished at the provider side is carried out implicitly through IND „Failure“ on all the AccoDataCRs in this AR_{SRT} at the consumer. Subsequently, the consumer shall reestablish the connections.

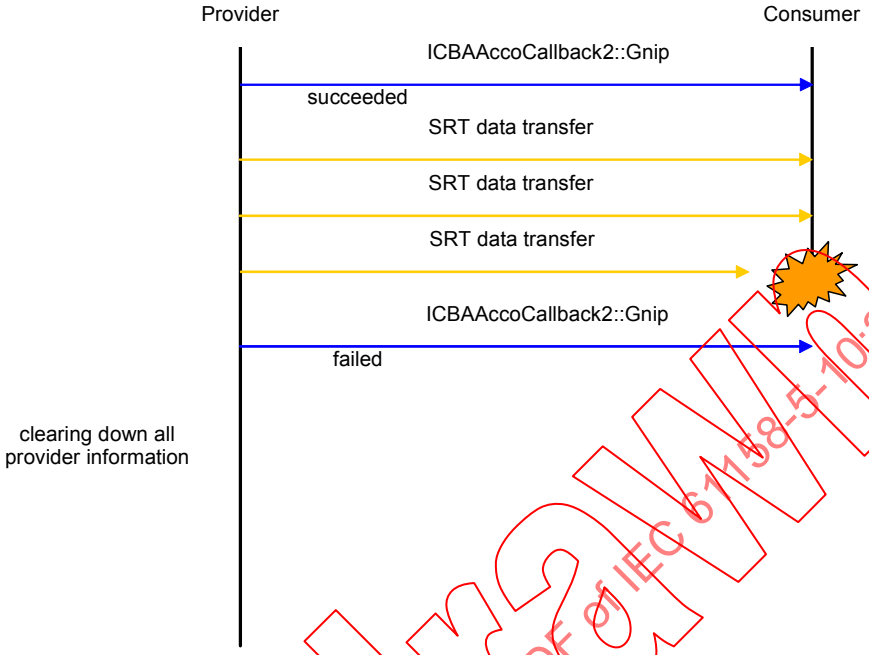


Figure 68 – Failure of the consumer

7.3.3.3.7.4.10 Unilateral communication failure recognition during productive operation

7.3.3.3.7.4.10.1 ORPC communication channel

The fault scenarios defined above and their reaction requires a unilateral recognition of a communication failure to be discussed separately. A unilateral recognition can occur due to a transient router failure or communication fault (poor connector contact, ...).

The provider discards all connections to the consumer concerned when it detects a communication failure. If the consumer subsequently – when communication is back to normal – tests the connections via the Ping service or the GetConnectionData service, the provider shall return a negative acknowledgement with CBA_S_NOCONNECTION if it does not maintain any connections to the nominated consumer. The consumer then follows the standard procedure for establishing all its connections.

A special form of this scenario results if - due to unilateral recognition of a communication failure - the provider discarded its connections. This consumer establishes subsequently before its next Ping or GetConnectionData new connections via the Connect2 service (ACCO Server interface). The subsequent Ping resp. GetConnectionData service would supply S_OK since the ping does not check the quantity of the connections, but whether or not there are any connections to the consumer.

The result value "FirstConnect" at the Connect2 service (ACCO Server interface) exists for this purpose. It shows the consumer that the established connections are the first ones the provider has from this consumer. Depending on the expectation of the consumer, it can recognize this situation and follow the normal error scenario (i.e. DisconnectMe service and new establishment of all connections). If, for example, the consumer thinks that connections have already been established in this provider, the Connect2 service shall return FirstConnect == false. If the value returned does not match the expectations, the error state of all connections regarding this provider shall be set to E_FAIL until reestablished.

7.3.3.3.7.4.10.2 RT communication channel

The fault scenarios defined above and their reaction requires a unilateral recognition of a communication failure to be discussed separately. A unilateral recognition can occur due to a transient switch failure or communication fault (poor connector contact ...).

The provider shall discard all connections, i.e. the complete AR_{SRT} to the consumer concerned when it detects a communication failure via the periodic invocation of the Gnip service (ACCO Callback interface). The consumer subsequently detects communication failures via indications by RT to all its AccoDataCRs in this AR_{SRT}. The consumer then shall follow the standard procedure for establishing all its connections.

NOTE The following scenario is not detected: (1) The engineering system clears the last connection to a specific provider via the RemoveConnections servive (ACCO Management interface). (2) The consumer invokes the Disconnect service (ACCO Server SRT interface) to the provider, whilst the communication is disturbed directly at the consumer's interface (carrier lost); communication to the provider is therefore instantly rejected. (3) After 10 seconds the consumer issues a new connect attempt, also resulting in a communication error. (4) Consumer expects that the provider drops its connections automatically. If after step (4) the network cable is plugged in again at the consumer and the provider has not invoked the Gnip service during step (2) – (4), the communication error is only detected by the consumer. The provider will continue to send data either until the next connection is established by the same consumer, until a communication failure is detected on invoking the Gnip service or until power-off of the provider. The engineering system is not able to detect this situation.

7.3.3.3.7.4.11 Failure of the provider during negotiation

If the communication to the provider fails while the consumer negotiates changes to the connections with the provider (set up connections, connection state change or clear connections), the consumer expects the failure of the provider. This procedure becomes effective e.g. after a negative confirmation of Connect2 (ACCO Server interface), ConnectCR (ACCO Server SRT interface) or Connect (ACCO Server SRT interface)).

The consumer shall assume all connections as not established, since in case of a failure the provider drops all connections established. The consumer then shall try to connect (see 7.3.3.3.6.2.3.2) in intervals depending on the specific error (see 7.3.3.3.7.4.2 for details).

Once the provider is available again, all connections at the provider's side shall be cleared by invoking the DisconnectMe service of the ACCO Server interface resp. the ACCO Server SRT interface, to clear situations in which the provider has not yet detected a communication failure.

Subsequently, the connections shall be negotiated with the provider to reflect the state currently on the consumer.

– In case of the ORPC communication channel the Connect2 service of the ACCO Server interface (typically multiple calls) is used.
– In case of the RT communication channel the ConnectCR and Connect service of the ACCO Server SRT interface are used to establish the AccoDataCRs (typically multiple sequences; see 7.3.3.3.5.3.4.3 for details).

If connections are to be removed, the resources for the connections in the consumer may only be released after the connection data in the provider has been cleared up. If the provider was not notified of the failure of the communication relationship between provider and consumer, it could at any time receive data for the specific (first cleared and subsequently re-assigned) item and distribute connection data to a wrong RT-Auto object. This procedure has a disadvantage that shall be looked separately: A connection can only be cleared on the consumer when the provider participates in this process. Connection „corpses" at the consumer side could be left over if a provider is removed completely from the distributed automation system without having cleared the connections upfront.

This situation is solved by the following means: If the last connections are cleared from the consumer and the consumer is not able to propagate the Disconnect to the provider, the consumer shall make a new connect attempt in intervals depending on the specific error (see

7.3.3.3.7.4.2 for details) to this very provider, and shall invoke the DisconnectMe service, follows also the mechanism described above. If this connect attempt fails, the consumer can safely assume that the provider himself detects a communication failure. The consumer can therefore drop the connections on his side and release the resources.

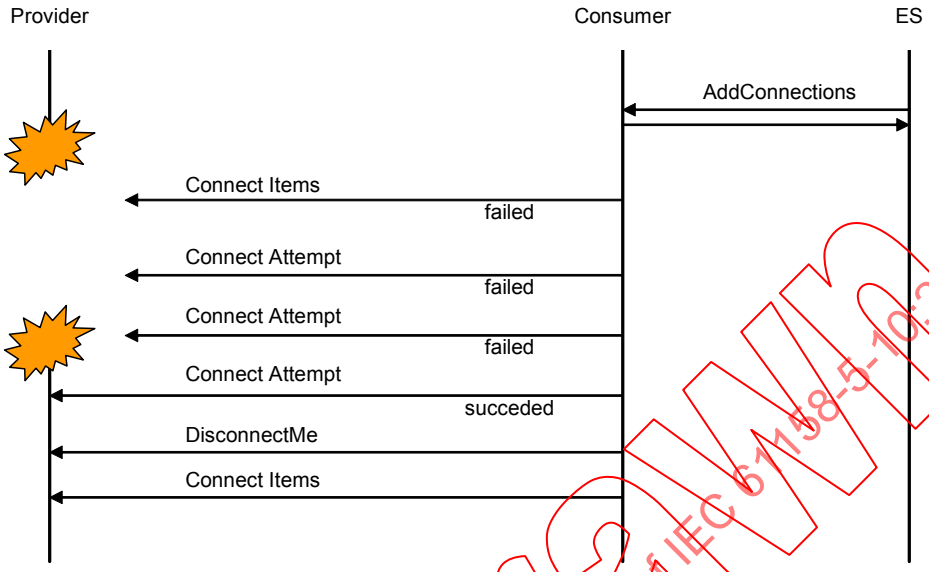


Figure 69 – Failure of the provider when setting up connections

7.3.3.3.7.4.12 Transmission of garbled data

If the consumer is not able to decode the transmitted connection data, he shall apply substitute values to all connection sinks delivered by this provider.

– In the ORPC communication channel, he shall return an error code to the provider and apply substitute values to all connection sinks delivered by this provider. The provider itself shall drop as reaction to the error code all connections established.
– In the RT communication channel all consumer resources regarding this provider shall be released.

Then all connections at the provider's side shall be cleared by calling the DisconnectMe service (ACCO Server interface resp. ACCO Server SRT interface). Subsequently, the connections shall be negotiated with the provider to reflect the state currently on the consumer.

– In case of the ORPC communication channel the Connect2 service of the ACCO Server interface (typically multiple calls) is used.
– In case of the RT communication channel the ConnectCR and Connect service of the ACCO Server SRT interface are used to establish the AccoDataCRs (typically multiple sequences; see 7.3.3.3.5.3.4.3 for details).

The Connect2 service (ACCO Server interface) and the Connect service (ACCO Server SRT interface) shall safeguard the data types, i.e. the provider and consumer have the same understanding of the data that needs to be exchanged.

7.3.3.3.8 Connection diagnosis

Like device diagnosis, different information levels exist for connection diagnosis as shown in Figure 70.

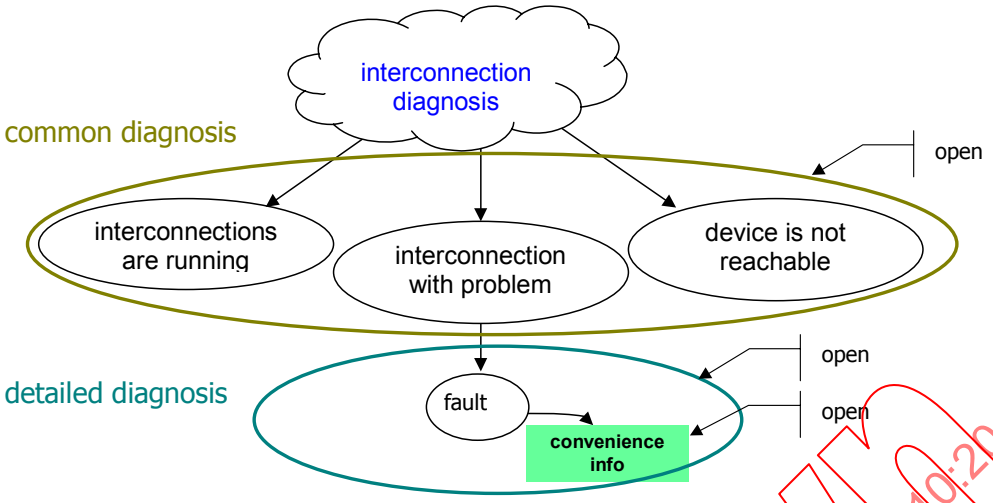


Figure 70 – Information levels

Status model for the common diagnosis

The common diagnosis of the ACCO ASE is shown in Figure 71.

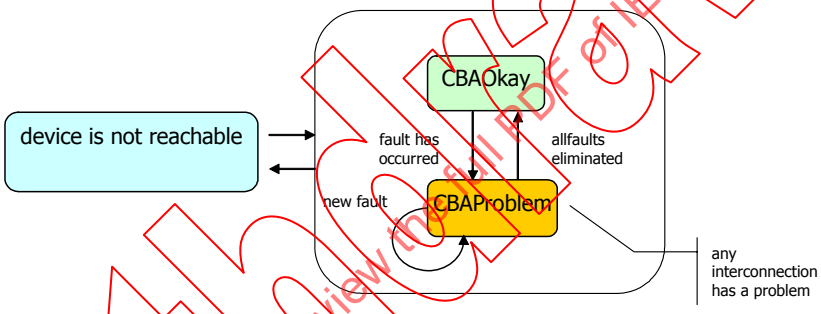


Figure 71 – ACCO ASE status model for the common diagnosis

The ACCO ASE common diagnosis uses the same interfaces that are used for the common diagnosis of the device (i.e. synchronous and asynchronous variant). Furthermore, the same rules apply to the distribution of the asynchronous notifications.

The common diagnosis states CBAUnknown and CBANonExistent are never visible on the GroupError of the ACCO object. They are only visible at the GroupError of a Proxy LDev object.

The GroupError of the ACCO object may be latched also as a part of the (connectable) system property “StateCollection”.

Status model for the detailed diagnosis

The representations of the connections in a dynamic diagram require information about the state of a specific connection to be obtained in order to be able to graphically edit this information according to the requirements. For the ACCO ASE, this information is understood under detailed diagnosis.

The Figure 72 shows the status model of a connection for the detailed diagnosis:

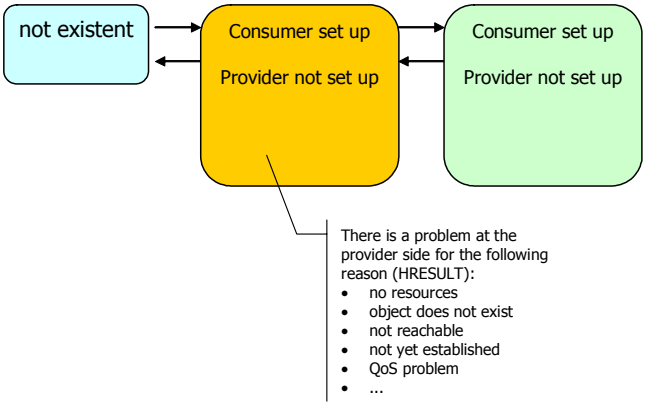


Figure 72 – ACCO ASE status model for the detailed diagnosis

Due to the ACCO ASE definition and its response to faults, the status model can easily be selected:

– Faults that occur during the setup of the consumer (no resources or object cannot be found) are returned synchronously when the connection is established via the AddConnections service. In this case, the connection is not yet established and remains in the "not existent" state.

– Faults that occur during the setup of the provider (no resources, object cannot be found, communication errors) are returned via detailed diagnosis and lead to a display of a group error. The consumer tries after a specific time depending on the specific error (see 7.3.3.3.7.4.2 for details) to purge the error by automatically reconnecting the connections – independent of the error. If an error is permanent, the user has to clear this connection explicitly to avoid traffic and in consequence to clear the group error.

  This applies also to local connections, i.e. connections with provider and consumer in the same logical device.

  In consequence even the error specifying a connection from an identifier to itself will also be reported via detailed diagnosis.

– If an object is cleared that is consumer of one or more connections, those associated connections are cleared automatically. Although this is a malfunction of the engineering system (an object shall only be cleared after all the connections to this object have been cleared), it avoids "corpses" that would probably never be cleared by the engineering system.

This is why the complete error information can be reduced to errors regarding the provider. These errors are represented by an error code (standard ORPC error code, i.e. HRESULT). The consumer of a connection consequently holds the current error information.

If error codes are returned from the property put function (for example if a buffer gets unpacked in the implementation of the OnDataChanged service), all are noted in the detailed diagnosis, whereas status codes (like CBA_S_VALUEBUFFERED) are dropped.

Table 181 shows the possible faults. The table shows also in the column group error, whether a specific fault results in displaying a group error on the ACCO ASE. A group error is displayed, if one or more connections have an error code with severity set to "error" (characterized by E_ in the error code's literal). The group error is cleared, if none of the connections have such an error code.

Table 181 – Error codes for the ACCO ASE detailed diagnosis

Fault description	Error code	GroupError
The provider's or the consumer's resources prove insufficient for establishing or operating the connection.	E_OUTOFMEMORY	yes
The communication failed due to any other reason. The value of the flag "FirstConnect" returned from the Connect2 service (ACCO Server interface) or the ConnectCR service (ACCO Server SRT interface) was not expected.	E_FAIL	yes
The target item could not be written by the consumer due to limitations of the implementation.	E_INVALIDARG	yes
The device is not supporting the complete spectrum of DATE (01.01.100 – 31.12.9999) and the date provided falls into that non supported area.	CBA_E_TIMEVALUEUNSUPPORTED	yes
The Source RT-Auto object cannot be found in the provider.	CBA_E_UNKNOWNOBJECT	yes
The Source Item cannot be found in the provider.	CBA_E_UNKNOWNMEMBER	yes
The Source Item has a different data type.	CBA_E_TYPEMISMATCH	yes
The provider cannot provide the data with the necessary QoS.	CBA_E_QOSTYPEUNSUPPORTED CBA_E_QOSVALUEUNSUPPORTED	yes
Communication error.	RPC_* (e.g. RPC_S_SERVER_UNAVAILABLE)	yes
It is not allowed to specify a connection from an identifier to itself.	CBA_E_INVALIDCONNECTION	yes
The data exceeds limits of the corresponding data type.	CBA_E_LIMITVIOLATION	yes
The request exceeds the resources according to their count.	CBA_E_COUNTEXCEEDED	yes
The request exceeds the resources according to their size.	CBA_E_SIZEEXCEEDED	yes
The request exceeds the resources according to the AccoPairs.	CBA_E_OUTOFACCOPAIRS	yes
The request exceeds the resources according to the partner ACCO objects.	CBA_E_OUTOFPARTNERACCOS	yes
The request exceeds the restriction of an AR_{SRT} to one AccoDataCR per QoS.	CBA_E_FRAMECOUNTUNSUPPORTED	yes
The remote station failed regarding the data transfer.	CBA_E_STATIONFAILURE	yes
The network card's current link status is unsuitable for this protocol.	CBA_E_LINKFAILURE	yes
The size of the item exceeds the upper limit for RT.	CBA_E_ITEMTOOLARGE	yes
The connection is not yet been established completely.	CBA_S_ESTABLISHING	no
Connection Data is currently not received.	CBA_S_NOCONNECTIONDATA	no

The synchronous form of the detailed diagnosis is integrated into the existing ACCO class. For devices with Base Object Version less than 2 the services GetIDs and GetConnections are provided at the ACCO Management interface. For devices with Base Object Version greater or equal to 2 the services GetConsID and GetConsConnections of the ACCO Management interface shall be used instead.

Diagnosis information from the consumer

The services DiagConsConnections and GetDiagnosis of the ACCO Management interface are provided for the extended detailed diagnosis information from the consumer.

Diagnosis information from the provider

The services GetProvsIDs, GetProvConnections and GetDiagnosis of the ACCO Management interface are provided for the extended detailed diagnosis information from the provider.

7.3.3.3.9 Synchronous Read / Write

7.3.3.3.9.1 Overview

Custom interfaces of the RT-Auto class permit synchronous access to data. The corresponding property access services can be used for reading or writing the value of a property.

The ACCO Sync interface enables a client to consistently access value and quality code. Thus the following services are provided:

– ReadItems to read consistently value and quality code for a set of items.
– WriteItems to write value with default quality code for a set of items.
– WriteItemsQCD to write consistently value and quality code for a set of items.

The services WriteItems and ReadItems support additionally access to single elements of structures or arrays.

7.3.3.3.9.2 Synchronous access

Custom interfaces of the RT-Auto class permit access to data. Their corresponding property access services can be used for reading or writing the value of a property. The quality code cannot directly be retrieved from a QC-unaware RT-Auto object since it is generated and managed by the ACCO ASE. Only an inconsistent access (since two operations are required) is possible for a QC-aware RT-Auto object.

The ACCO Class provides with its services on the ACCO Sync interface a consistent access to the value and quality code. This interface enables a client to perform consistent read and/or write access procedures to value, Quality code and time stamp - irrespective of the QC-awareness of the RT-Auto object. The interface permits symbolic access to the values.

A plaintext identifier - analog to the identification of the source or sink of a connection - is used for the symbolic identification of the values. The identifier („Object.Member”) shall be defined as described in 7.3.3.3.4.3.2.

7.3.3.4 ACCO Service specification

7.3.3.4.1 Interface ACCO Management

7.3.3.4.1.1 AddConnections

The AddConnection service adds one or multiple connection records to the ACCO object. Table 182 shows the parameters of the service.

NOTE If connections on structures are added, the substitute value is to be supplied as a VARIANT stating a SAFEARRAY of unsigned char containing first the type description followed by the content as defined in 7.3.3.4.3.1.

Table 182 – AddConnections (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Provider	M	M(=)		
QoSType	M	M(=)		
QoSValue	M	M(=)		
State	M	M(=)		
Count	M	M(=)		
List of pAddConnectionIn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppAddConnectionOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Provider
This parameter contains the identifier of the LDev of the provider.

Parameter type: LPWSTR

QoSType
This parameter contains the designation of the QoS type for all connections.

Parameter type: unsigned short

QoSValue
This parameter contains the designation of the QoS value for all connections.

Parameter type: unsigned short

State
This parameter contains the initial state of the connection; i.e. data transmission immediately and/or after the invocation of SetActivationState service.

Parameter type: VARIANT_BOOL

Count
This parameter contains the number of connections that are to be added.

Parameter type: unsigned long

List of pAddConnectionIn
This parameter contains the connection information. The number of elements is determined by the parameter Count.

Parameter type: ADDCONNECTIONIN

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppAddConnectionOut
This parameter contains the connection results. The number of elements is determined by the parameter Count.

Parameter type: ADDCONNECTIONOUT

Allowed values for ErrorState (hresult = S_OK): S_OK

Allowed values for ErrorState (hresult = S_FALSE): S_OK, CBA_E_MALFORMED, CBA_E_UNKNOWNOBJECT, CBA_E_UNKNOWNMEMBER, CBA_E_INUSE, CBA_E_INVALIDSUBSTITUTE, CBA_E_INVALIDEPSILON, CBA_E_INVALIDENUMVALUE, CBA_E_DISCONNECTRUNNING, CBA_E_CAPACITYEXCEEDED, CBA_E_COUNTEXCEEDED, CBA_E_SIZEEXCEEDED, CBA_E_ITEMTOOLARGE, CBA_E_NOTROUTABLE, E_ACCESSDENIED, E_POINTER, E_UNEXPECTED

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_MALFORMED, CBA_E_QOSTYPENOTAPPLICABLE, CBA_E_QOSTYPEUNSUPPORTED, CBA_E_QOSVALUEUNSUPPORTED, CBA_E_PERSISTRUNNING, CBA_E_DISCONNECTRUNNING, CBA_E_OUTOFPARTNERACCOS, CBA_E_OUTOFACCOPAIRS, E_OUTOFMEMORY, E_FAIL, E_UNEXPECTED, E_POINTER, RPC_*

7.3.3.4.1.2 RemoveConnections

The RemoveConnection service removes one or multiple connection records from the ConsumerIDs that were obtained via AddConnections. Table 183 shows the parameters of the service.

Table 183 – RemoveConnections (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pConsumerID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections that are to be removed.

Parameter type: unsigned long

List of pConsumerID
This parameter contains the ConsumerIDs of the connections. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the connection specific faults. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_E_INVALIDID, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_PERSISTRUNNING, E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.3 ClearConnections

The ClearConnections service clears all connections in the ACCO object. Table 184 shows the parameters of the service.

Table 184 – ClearConnections (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request..

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_PERSISTRUNNING, E_OUTOFMEMORY, E_FAIL, RPC_*

7.3.3.4.1.4 SetActivationState

The SetActivationState service influences the activation state (enable or disable) of one or multiple connections. Table 185 shows the parameters of the service.

Table 185 – SetActivationState (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
State	M	M(=)		
Count	M	M(=)		
List of pConsumerID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

State
This parameter contains the state of the connection (i.e. whether data shall be transferred for this connection).

Parameter type: VARIANT_BOOL

Count
This parameter contains the number of connections that are to be activated or deactivated.

Parameter type: unsigned long

List of pConsumerID
This parameter contains the ConsumerIDs of the connections. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the connection specific faults. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_E_INVALIDID, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_PERSISTRUNNING, E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.5 GetInfo

The GetInfo service reads the maximum number of possible and currently used connections. The maximum number of possible connections is based on the current resource situation and the assumption, that all following connections loaded are of scalar type, i.e. from VARIANT_BOOL up to double.

NOTE This service is depreciated for devices with Base Object Version 2 or later. Use the statistic for performance characteristics supplied via the GetDiagnosis service of the ICBAAccoMgt2 interface.

Table 186 shows the parameters of the service.

Table 186 – GetInfo (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pMax			M	M(=)
pCurCnt			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pMax
This parameter contains the maximum possible number of connections. If the ACCO object is unable to provide a reasonable estimation, 0xFFFFFFFF is returned instead.

Parameter type: unsigned long

pCurCnt
This parameter contains the number of currently used connections.

Parameter type: unsigned long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.6 GetIDs

The GetIDs service reads the major connection information.

NOTE 1 This service is depreciated for devices with Base Object Version 2 or later. Use the GetConsIDs service of the ICBAAccoMgt2 interface instead.

NOTE 2 The symbolic information of the connections can be determined by converting the obtained ConsumerIDs via the GetConnections service into the complete connection information. The symbolic information of the connection remains constant as long as the „Version" information of a connection does not change. Thus, invoking the GetIDs service proves sufficient for determining the state.

Table 187 shows the parameters of the service.

Table 187 – GetIDs (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pCount			M	M(=)
List of ppGetIDOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pCount
This parameter contains the number of connections.

Parameter type: unsigned long

List of ppGetIDOut
This parameter contains the connection specific properties. The number of elements is determined by the parameter pCount.

Parameter type: GETIDOUT

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.7 GetConnections

The GetConnections service reads all connection information.

NOTE 1 This service is depreciated for devices with Base Object Version 2 or later. Use the GetConsConnections service of the ICBAAccoMgt2 interface instead.

NOTE 2 The GetConnections service returns names regarding upper and lower case as follows:The ConsumerItem is returned according to its internal notation (like returned from the BrowseItems2 service).The ProviderName is returned according to its first use with the AddConnections or Connect service.

NOTE 3 If connections on structures are returned, the substitute value is to be supplied as a VARIANT stating a SAFEARRAY of unsigned char containing first the type description followed by the content as defined in 7.3.3.4.3.1.

Table 188 shows the parameters of the service.

Table 188 – GetConnections (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pConsumerID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppGetConnectionOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections.

Parameter type: unsigned long

List of pConsumerID
This parameter contains the ConsumerIDs of all connections of this ACCO object. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

NOTE 4 S_FALSE is returned, if the service failed on one or more specific connections. This partial errors are contained in the ErrorState fields of the parameter ppGetConnectionOut.

List of ppGetConnectionOut
This parameter contains the connection specific properties. The number of elements is determined by the parameter Count.

Parameter type: GETCONNECTIONOUT

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.8 ReviseQoS

The ReviseQoS service allows to verify a QoS value. It informs about the possible QoS, both type and Value, supported by an ACCO object. Table 189 shows the parameters of the service.

Table 189 – ReviseQoS (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
RTAuto	M	M(=)		
QoSType	M	M(=)		
QoSValue	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pRevisedQoSValue			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

RTAuto
This parameter contains the identifier of the RT-Auto object.

Parameter type: LPWSTR

QoSType
This parameter contains the designation of the QoS type for all connections.

Parameter type: unsigned short

Allowed values: according to Table 171

QoSValue
This parameter contains the designation of the QoS value for all connections.

Parameter type: unsigned short

Allowed values: according to Table 171

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pRevisedQoSValue
This parameter contains revised additional information.

Parameter type: unsigned short

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_MALFORMED, CBA_E_UNKNOWNOBJECT, CBA_E_QOSTYPEUNSUPPORTED, E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.9 get_PingFactor

The get_PingFactor service reads the current PingFactor that is used for determining the maximum dead time of a provider. This service returns the attribute PingFactor. After the PingFactor gets changed via the put_PingFactor service up to reflecting that change to the persistence data base, this service returns CBA_S_PERSISTPENDING to inform the caller. Table 190 shows the parameters of the service.

Table 190 – get_PingFactor (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the PingFactor.

Parameter type: unsigned short

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_S_PERSISTPENDING, E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.10 put_PingFactor

The put_PingFactor service sets the ping factor that is used for determining the maximum dead time of a provider. Setting the ping factor to 0 disables pinging. This service writes the attribute PingFactor. The change shall be reflected to the persistence data base via the Save service. Table 191 shows the parameters of the service.

Table 191 – put_PingFactor (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
NewVal	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

NewVal
This parameter contains the dead time within which a message is expected from a provider (PingFactor * minimum QoS).

Parameter type: unsigned short

Allowed values: see Table 168.

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.11 get_CDBCookie

The get_CDBCookie service reads the current Configuration Data Base change indicator. It changes if the Configuration Data Base changes. Table 192 shows the parameters of the service.

NOTE This service is depreciated for devices with Base Object Version 2 or later. Use the system variable for the ACCO Stamp (see 7.3.3.3.4.12) instead.

Table 192 – get_CDBCookie (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the Configuration Data Base change indicator.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_NOTIMPL, E_POINTER, RPC_*

7.3.3.4.1.12 GetConsIDs

The GetConsIDs service reads the current consumer connection IDs. The complete connection information held on this consumer can be determined by invoking the GetConsConnections service with the obtained ConsumerID. Table 193 shows the parameters of the service.

Table 193 – GetConsIDs (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pCount			M	M(=)
List of ppConsumerID			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pCount
This parameter contains the number of consumer IDs.

Parameter type: unsigned long

List of ppConsumerID
This parameter contains the consumer IDs. The number of elements is determined by the parameter pCount.

Parameter type: unsigned long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.13 GetConsConnections

The GetConsConnections service reads the consumer connection information.

NOTE 1 The GetConsConnections service returns names regarding upper and lower case as follows: The ConsumerItem is returned according to its internal notation (like returned from the BrowseItems2 service). The ProviderName is returned according to its first use with AddConnections, Connect or Connect2 service.

NOTE 2 If connections on structures are returned, the substitute value is to be supplied as a VARIANT stating a SAFEARRAY of unsigned char containing first the type description followed by the content as defined in 7.3.3.4.3.1.

Table 194 shows the parameters of the service.

Table 194 – GetConsConnections (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pConsumerID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppGetConsConnOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections.

Parameter type: unsigned long

List of pConsumerID
This parameter contains the ConsumerIDs of the required connection information. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppGetConsConnOut
This parameter contains the consumer connection information. The number of elements is determined by the parameter Count.

Parameter type: GETCONSCONNOUT

Allowed values for PartialResult(hresult = S_OK): S_OK

Allowed values for PartialResult(hresult = S_FALSE): S_OK, CBA_E_INVALIDID, E_OUTOFMEMORY, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.14 DiagConsConnections

The DiagConsConnections service reads the consumer connection diagnosis.

Table 195 shows the parameters of the service.

Table 195 – DiagConsConnections (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pConsumerID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppDiagConsConnOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections.

Parameter type: unsigned long

List of pConsumerID
This parameter contains the ConsumerIDs to be diagnosed. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppDiagConsConnOut
This parameter contains the consumer connection diagnosis. The number of elements is determined by the parameter Count.

Parameter type: DIAGCONSCONNOUT

Allowed values for PartialResult(hresult = S_OK): S_OK

Allowed values for PartialResult(hresult = S_FALSE): S_OK, CBA_E_INVALIDID, E_OUTOFMEMORY, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.15 GetProvIDs

The GetProvIDs service reads the current provider connection IDs. The partial connection information held on this provider can be determined by invoking the GetProvConnections service with the obtained ProviderID. Table 196 shows the parameters of the service.

Table 196 – GetProvIDs (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pCount			M	M(=)
List of ppProviderID			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pCount
This parameter contains the number of provider IDs.

Parameter type: unsigned long

List of ppProviderID
This parameter contains the provider IDs. The number of elements is determined by the parameter pCount.

Parameter type: unsigned long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.16 GetProvConnections

The GetProvConnections service reads all connection information the provider holds over specified connections.

NOTE The GetProvsConnections service returns names regarding upper and lower case as follows: The ProviderItem is returned according to its internal notation (like returned from the BrowseItems2 service). The ConsumerName is returned according to its first use with Connect or Connect2 service.

Table 197 shows the parameters of the service.

Table 197 – GetProvConnections (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pProviderID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppGetProvConnOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections.

Parameter type: unsigned long

List of pConsumerID
This parameter contains the ProviderIDs of the required connection information. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppGetProvConnOut
This parameter contains the provider connection information. The number of elements is determined by the parameter Count.

Parameter type: GETPROVCONNOUT

Allowed values for PartialResult(hresult = S_OK): S_OK

Allowed values for PartialResult(hresult = S_FALSE): S_OK, CBA_E_INVALIDID, E_OUTOFMEMORY, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.1.17 GetDiagnosis

The GetDiagnosis service reads detailed diagnosis information from both consumer and provider. The method uses a request triggered mechanism with filter information. Table 198 shows the parameters of the service.

Table 198 – GetDiagnosis (ACCO Management interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Request	M	M(=)		
InLength	M	M(=)		
pInBuffer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pOutLength			M	M(=)
ppOutBuffer			U	U(=)
Diagnosis Function Directory			S	S(=)
IDCount			M	M(=)
List of RequestID			M	M(=)
Performance Characteristics			S	S(=)
General Parameters			M	M(=)
CntLDevsMax			M	M(=)
CntLDevsCur			M	M(=)
CntRTAutosMax			M	M(=)
CntRTAutosCur			M	M(=)
CntPropertiesMax			M	M(=)
CntPropertiesCur			M	M(=)
SizePropertyMax			M	M(=)
SizeTypeDescriptionMax			M	M(=)
SizeTypeDescriptionCur			M	M(=)
SizePropertiesInMax			M	M(=)
SizePropertiesInCur			M	M(=)
SizePropertiesOutMax			M	M(=)
SizePropertiesOutCur			M	M(=)
Connections Parameters			M	M(=)
CntPartnerAccosMax			M	M(=)
CntPartnerAccosCur			M	M(=)
CntAccoPairsMax			M	M(=)
CntAccoPairsCur			M	M(=)
Constant Connections Parameters			M	M(=)
CntConnectionsMax			M	M(=)
CntConnectionsCur			M	M(=)
SizeConnectionsMax			M	M(=)
SizeConnectionsCur			M	M(=)
Local Connections Parameters			M	M(=)
CntConnectionsMax			M	M(=)
CntConnectionsCur			M	M(=)
SizeConnectionsMax			M	M(=)
SizeConnectionsCur			M	M(=)
RT Connections Parameters			M	M(=)
MinQoSValue			M	M(=)
CntConsConnsMax			M	M(=)
CntConsConnsCur			M	M(=)
SizeConsConnsMax			M	M(=)
SizeConsConnsCur			M	M(=)
CntConsFramesMax			M	M(=)
CntConsFramesCur			M	M(=)
CntProvConnsMax			M	M(=)
CntProvConnsCur			M	M(=)
SizeProvConnsMax			M	M(=)
SizeProvConnsCur			M	M(=)
CntProvFramesMax			M	M(=)
CntProvFramesCur			M	M(=)
ORPC Productive Data Connections Parameters			M	M(=)
MinQoSValue			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
CntConsConnsMax			M	M(=)
CntConsConnsCur			M	M(=)
SizeConsConnsMax			M	M(=)
SizeConsConnsCur			M	M(=)
CntProvConnsMax			M	M(=)
CntProvConnsCur			M	M(=)
SizeProvConnsMax			M	M(=)
SizeProvConnsCur			M	M(=)
ORPC HMI Connections Parameters			M	M(=)
MinQoSValue			M	M(=)
CntProvConnsMax			M	M(=)
CntProvConnsCur			M	M(=)
SizeProvConnsMax			M	M(=)
SizeProvConnsCur			M	M(=)
ORPC Status Connections Parameters			M	M(=)
CntProvConnsMax			M	M(=)
CntProvConnsCur			M	M(=)
SizeProvConnsMax			M	M(=)
SizeProvConnsCur			M	M(=)
Further Device Parameters			M	M(=)
CntAdvisesMax			M	M(=)
CntAdvisesCur			M	M(=)
Consumer Communication Event Statistic			S	S(=)
WatchTP			M	M(=)
ORPCFailureCnt			M	M(=)
SrtFrameFailureCnt			M	M(=)
ORPCReceivedNotGoodCnt			M	M(=)
Provider Communication Event Statistic			S	S(=)
WatchTP			M	M(=)
ORPCQoSViolationCnt			M	M(=)
SrtDropCnt			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

Request
This parameter contains the information which diagnosis function is requested.

Parameter type: unsigned long

Allowed values: 0x00000000 – 0x7FFFFFFF for centralized administrated diagnosis functions, 0x80000000 – 0xFFFFFFFF for vendor specific diagnosis functions

The predefined values for centralized administrated diagnosis functions are shown in Table 199

Table 199 – Request

Value	RequestID	Description
0	DIAGNOSIS_FUNCTION_DIRECTORY	The Diagnosis Function Directory shall be read.
0x1000	PERFORMANCE_CHARACTERISTICS	The Performance Characteristics shall be read.
0x2000	STATISTICS_RESET	All values of Consumer Communication Events Statistic and Provider Communication Events Statistic shall be set to 0.
0x3000	CONSUMER_COMMUNICATION_EVENT_STATISTIC	The Consumer Communication Events Statistic shall be read.
0x4000	PROVIDER_COMMUNICATION_EVENT_STATISTIC	Provider Communication Events Statistic shall be read.

InLength
This parameter contains the overall length of the filter information in octets. For requesting of diagnosis data that are referenced in Table 199 it shall be set to 0.

Parameter type: unsigned long

pInBuffer
This parameter contains the filter information. The data shall be serialized according the used data types with least significant octet first and without any padding octets. For requesting of diagnosis data that are referenced in Table 199 it shall be empty.

Parameter type: OctetString

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

pOutLength
This parameter contains the overall length of the diagnosis data in octets.

Parameter type: unsigned long

ppOutBuffer
This parameter contains the diagnosis data. The data shall be serialized according the used data types with least significant octet first and without any padding octets.

Diagnosis Function Directory
This parameter contains the Request IDs of the centralized administrated diagnosis functions that are available for the device.

NOTE 1 Only identifiers of centralized administrated diagnosis functions should be returned. For vendor specific diagnosis functions a separate directory functionality should be implemented.

IDCount
This parameter contains the number of Request IDs.

Parameter type: unsigned long

List of RequestID
This parameter contains the Request IDs. The number of elements is determined by the parameter IDCount.

Parameter type: unsigned long

Performance Characteristics
This parameter contains the performance characteristics parameters regarding their initial and their current consumption. The parameter values contain always global information for the complete device, independent on which ACCO object it was called.

NOTE 2 „Supported communication protocols" is not a separate parameter but can be derived from the communication channel related sub parameters, e.g. if "RT Channel Parameters.CntConsFramesMax" is greater than 0, the device supports the RT communication channel.

General Parameters

This parameter contains information that appear offline when components are assigned to the device.

CntLDevsMax

This parameter contains the maximum number of LDev objects that can be applied in the device.

Parameter type: unsigned long

CntLDevsCur

This parameter contains the current number of LDev objects in the device.

Parameter type: unsigned long

CntRTAutosMax

This parameter contains the maximum number of RT-Auto objects that can be applied in the device.

Parameter type: unsigned long

CntRTAutosCur

This parameter contains the current number of RT-Auto objects in the device.

Parameter type: unsigned long

CntPropertiesMax

This parameter contains the maximum number of properties that can be applied in the device.

Parameter type: unsigned long

CntPropertiesCur

This parameter contains the current number of properties in the device.

Parameter type: unsigned long

SizePropertyMax

This parameter contains the maximum data size of a property that can be applied in the device.

Parameter type: unsigned long

SizeTypeDescriptionMax

This parameter contains the maximum size of all type descriptions that can be applied in the device.

Parameter type: unsigned long

SizeTypeDescriptionCur

This parameter contains the current size of all type descriptions in the device.

Parameter type: unsigned long

SizePropertiesInMax

This parameter contains the maximum data size for all input properties that can be applied in the device.

Parameter type: unsigned long

SizePropertiesInCur

This parameter contains the current data size for all input properties in the device.

Parameter type: unsigned long

SizePropertiesOutMax

This parameter contains the maximum data size for all output properties that can be applied in the device.

Parameter type: unsigned long

SizePropertiesOutCur
This parameter contains the current data size for all output properties in the device.

Parameter type: unsigned long

Connections Parameters
This parameter contains connection relations information.

CntPartnerAccosMax
This parameter contains the maximum number of partner ACCO objects (consumers and providers) for connections that can be applied in the device. Partners shall be counted without regard to their configured communication channels (ORPC, RT).

NOTE 3 Partner ACCO objects should be counted from the point of view of the device, i.e. the number of ACCO objects with which ACCO objects of the device have a relation. If different ACCO objects within an device have a relation with the same ACCO object, this counts as one partner ACCO object. Partner ACCO objects needed for local and constant connections should be implicitly reserved.

Parameter type: unsigned long

CntPartnerAccosCur
This parameter contains the current number of partner ACCO objects (consumers and providers) for connections in the device.

Parameter type: unsigned long

CntAccoPairsMax
This parameter contains the maximum number of ACCO pairs that can be applied in the device. An ACCO pair is a relation between two ACCO objects with respect to a communication channel that handles at least one connection. Connections shall be counted in accordance with their configured communication channels (ORPC, RT, CONSTANT).

NOTE 4 ACCO pairs are always counted from the point of view of the connected objects, i.e. the number of ACCO objects with which an object has a relation with respect to a channel. ORPC productive, ORPC HMI, RT, local and constant count as different communication channels.

Parameter type: unsigned long

CntAccoPairsCur
This parameter contains the current number of ACCO pairs in the device.

Parameter type: unsigned long

Constant Connections Parameters
This parameter contains constant communication channel related information.

CntConnectionsMax
This parameter contains the maximum number of connections with constants that can be applied in the device.

Parameter type: unsigned long

CntConnectionsCur
This parameter contains the current number of connections with constants in the device.

Parameter type: unsigned long

SizeConnectionsMax
This parameter contains the maximum sum of data sizes of all constant connections (regarded as a pool) that can be applied in the device.

Parameter type: unsigned long

SizeConnectionsCur
This parameter contains the current sum of data sizes of all constant connections in the device.

Parameter type: unsigned long

Local Connections Parameters
This parameter contains local communication channel related information.

CntConnectionsMax
This parameter contains the maximum number of local connections that can be applied in the device.

Parameter type: unsigned long

CntConnectionsCur
This parameter contains the current number of local connections in the device.

Parameter type: unsigned long

SizeConnectionsMax
This parameter contains the maximum sum of data sizes of all local connections (regarded as a pool) that can be applied in the device.

Parameter type: unsigned long

SizeConnectionsCur
This parameter contains the current sum of data sizes of all local connections in the device.

Parameter type: unsigned long

RT Connections Parameters
This parameter contains RT communication channel related information.

MinQoSValue
This parameter contains the minimum QoS Value for the RT communication channel in the device.

NOTE 5 After appropriate conversion of the minimum QoS Value to the RT cycle, this should be supported by ACP. The device should also support all QoS Values above this value.

Parameter type: unsigned long

CntConsConnsMax
This parameter contains the maximum number of RT connections as consumers that can be applied in the device.

Parameter type: unsigned long

CntConsConnsCur
This parameter contains the current number of RT connections as consumers in the device.

Parameter type: unsigned long

SizeConsConnsMax
This parameter contains the maximum sum of data sizes of all RT connections as consumers (productive data connections) that can be applied in the device.

Parameter type: unsigned long

SizeConsConnsCur
This parameter contains the current sum of data sizes of all RT connections as consumers (productive data connections) in the device.

Parameter type: unsigned long

CntConsFramesMax
This parameter contains the maximum number of RT frames as consumers that can be applied in the device.

Parameter type: unsigned long

CntConsFramesCur
This parameter contains the current number of RT frames as consumers in the device.

Parameter type: unsigned long

CntProvConnsMax
This parameter contains the maximum number of RT connections as providers that can be applied in the device.

Parameter type: unsigned long

CntProvConnsCur
This parameter contains the current number of RT connections as providers in the device.

Parameter type: unsigned long

SizeProvConnsMax
This parameter contains the maximum sum of data sizes of all RT connections as providers (productive data connections) that can be applied in the device.

Parameter type: unsigned long

SizeProvConnsCur
This parameter contains the current number of RT frames as providers in the device.

Parameter type: unsigned long

CntProvFramesMax
This parameter contains the maximum number of RT frames as providers that can be applied in the device.

Parameter type: unsigned long

CntProvFramesCurConsumer
This parameter contains the current number of RT frames as providers in the device.

Parameter type: unsigned long

ORPC Productive Data Connections Parameters
This parameter contains ORPC communication channel (productive data connections) related information.

MinQoSValue
This parameter contains the minimum QoS Value for the ORPC communication channel (productive data connections) in the device.

Parameter type: unsigned long

CntConsConnsMax
This parameter contains the maximum number of ORPC connections as consumers (productive data connections) that can be applied in the device.

Parameter type: unsigned long

CntConsConnsCur
This parameter contains the current number of ORPC connections as consumers (productive data connections) in the device.

Parameter type: unsigned long

SizeConsConnsMax
This parameter contains the maximum sum of data sizes of all ORPC connections as consumers (productive data connections) that can be applied in the device.

Parameter type: unsigned long

SizeConsConnsCur
This parameter contains the current sum of data sizes of all ORPC connections as consumers (productive data connections) in the device.

Parameter type: unsigned long

CntProvConnsMax
This parameter contains the maximum number of ORPC connections as providers (productive data connections) that can be applied in the device.

Parameter type: unsiged long

CntProvConnsCur
This parameter contains the current number of ORPC connections as providers (productive data connections) in the device.

Parameter type: unsigned long

SizeProvConnsMax
This parameter contains the maximum sum of data sizes of all ORPC connections as providers (productive data connections) that can be applied in the device.

Parameter type: unsigned long

SizeProvConnsCur
This parameter contains the current sum of data sizes of all ORPC connections as providers (productive data connections) in the device.

Parameter type: unsigned long

ORPC HMI Connection Parameters
This parameter contains ORPC communication channel (HMI connections) related information.

MinQoSValue
This parameter contains the minimum QoS Value for the ORPC communication channel (HMI connections) in the device.

Parameter type: unsigned long

CntProvConnsMax
This parameter contains the maximum number of ORPC connections as providers (HMI connections) that can be applied in the device.

Parameter type: unsigned long

CntProvConnsCur
This parameter contains the current number of ORPC connections as providers (HMI connections) in the device.

Parameter type: unsigned long

SizeProvConnsMax
This parameter contains the maximum sum of data sizes of all ORPC connections as providers (HMI connections) that can be applied in the device.

Parameter type: unsigned long

SizeProvConnsCur
This parameter contains the current sum of data sizes of all ORPC connections as providers (HMI connections) in the device.

Parameter type: unsigned long

ORPC Status Connection Parameters
This parameter contains ORPC communication channel (status connections) related information.

NOTE 6 A minimum QoS Value of 500 ms for ORPC status connections is defined, therefore no MinQoSValue sub parameter is given.

CntProvConnsMax
This parameter contains the maximum number of ORPC connections as providers (status connections) that can be applied in the device.

Parameter type: unsigned long

CntProvConnsCur
This parameter contains the current number of ORPC connections as providers (status connections) in the device.

Parameter type: unsigned long

SizeProvConnsMax
This parameter contains the maximum sum of data sizes of all ORPC connections as providers (status connections) that can be applied in the device.

Parameter type: unsigned long

SizeProvConnsCur
This parameter contains the current sum of data sizes of all ORPC connections as providers (status connections) in the device.

Parameter type: unsigned long

Further Device Parameters
This parameter contains further device information.

CntAdvisesMax
This parameter contains the maximum number of Advise events (State Advises and Group Error Advises) that can be handled simultaneously in the device.

Parameter type: unsigned long

CntAdvisesCur
This parameter contains the current number of Advise events (State Advises and Group Error Advises) that will be handled in the device.

Parameter type: unsigned long

Consumer Communication Event Statistic
This parameter contains an accumulated statistic regarding the communication events on the consumer. The connection errors of all ACCO objects in every LDev object are summed-up in the contained sub parameters. No persistency of the statistic values is provided. All parameter values can be reseted with RequestID STATISTICS_RESET.

WatchTP
This parameter contains the observation period in seconds since last reset.

Parameter type: unsigned long

ORPCFailureCnt
This parameter contains the number of error events regarding ORPC AccoPairs.

Parameter type: unsigned long

SrtFrameFailureCnt
This parameter contains the number of error events regarding RT Frames.

Parameter type: unsigned long

ORPCReceivedNotGoodCnt
This parameter contains the number of property granular with QC != Good.

Parameter type: unsigned long

Provider Communication Event Statistic
This parameter contains an accumulated statistic regarding the communication events on the provider. The connection errors of all ACCO objects in every LDev object are summed-up in the contained sub parameters. No persistency of the statistic values is provided. All parameter values can be reseted with RequestID STATISTICS_RESET.

WatchTP
This parameter contains the observation period in seconds since last reset.

Parameter type: unsigned long

ORPCQoSViolationCnt
This parameter contains the number of error events regarding QoS Violations.

Parameter type: unsigned long

SrtDropCnt
This parameter contains the number of IND ACP_STATE_PROV_DROP.

Parameter type: unsigned long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoMgt2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_INVALIDID, E_POINTER, RPC_*

7.3.3.4.2 Interface ACCO Server

7.3.3.4.2.1 Connect

The Connect service adds one or multiple connection sources to the ACCO object that acts as the connection provider. Table 200 shows the parameters of the service.

NOTE 1 This service is depreciated for devices with Base Object Version 2 or later. Use the service Connect2 instead.

NOTE 2 The service request is rejected for the QoSType CONSTANT and is rejected for the QoSType ORPC ST on non system variables.

If a valid reference to a callback interface is supplied with the parameter pICBAAccoCallback, connection data is transferred via push mode; i.e. using the OnDataChanged service. If an invalid reference (NULL) is supplied, connection data is transferred via pull mode; i.e. using the GetConnectionData service.

Table 200 – Connect (ACCO Server interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Consumer	M	M(=)		
QoSType	M	M(=)		
QoSValue	M	M(=)		
State	M	M(=)		
pICBAAccoCallback	M	M(=)		
Count	M	M(=)		
List of pConnectIn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pFirstConnect			M	M(=)
List of ppConnectOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

Consumer
This parameter contains the identifier of the consumer's LDev object in the form of "PDev!LDev".

Parameter type: LPWSTR

QoSType
This parameter contains the designation of the QoS type for all connections.

Parameter type: unsigned short

QoSValue
This parameter contains the designation of the QoS value for all connections.

Parameter type: unsigned short

State
This parameter contains the initial state of the connection; i.e. data transmission immediately and/or after the invocation of the SetActivationState service.

Parameter type: VARIANT_BOOL

pICBAAccoCallback
This parameter contains the reference to the callback interface that is used for transferring connection data.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections that are to be connected.

Parameter type: unsigned long

List of pConnectIn
This parameter contains the connection information. The number of elements is determined by the parameter Count.

Parameter type: CONNECTIN

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

pFirstConnect
This parameter contains information whether this is the first established connection to this consumer.

Parameter type: VARIANT_BOOL

List of ppConnectOut
This parameter contains the connection specific results. The number of elements is determined by the parameter Count.

Parameter type: CONNECTOUT

Allowed values for ErrorState (hresult = S_OK): S_OK

Allowed values for ErrorState (hresult = S_FALSE): S_OK, CBA_E_MALFORMED, CBA_E_UNKNOWNOBJECT, CBA_E_UNKNOWNMEMBER, CBA_E_INUSE, CBA_E_QOSTYPEUNSUPPORTED, CBA_E_QOSVALUEUNSUPPORTED, CBA_E_INVALIDEPSILON, CBA_E_TYPEMISMATCH, CBA_E_COUNTEXCEEDED, CBA_E_SIZEEXCEEDED, E_POINTER, E_ACCESSDENIED, E_UNEXPECTED

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Allowed values: CBA_E_MALFORMED, CBA_E_NOTAPPLICABLE, CBA_E_OUTOFPARTNERACCOS, CBA_E_OUTOFACCOPAIRS, E_OUTOFMEMORY, E_FAIL, E_UNEXPECTED, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.2.2 Disconnect

The Disconnect service clears one or multiple connection blocks from the ProviderIDs that were obtained via the Connect service. Table 201 shows the parameters of the service.

Table 201 – Disconnect (ACCO Server interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pProviderID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections that are to be disconnected.

Parameter type: unsigned long

List of pProviderID
This parameter contains the provider IDs of the connections. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the connection specific faults. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_E_NOTAPPLICABLE, CBA_E_INVALIDID

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.3.4.2.3 DisconnectMe

The DisconnectMe service clears all connection blocks of a consumer. It is typically used by the consumer in its startup in order to clear all existing connections and to reestablish them afterwards.Table 202 shows the parameters of the service.

Table 202 – DisconnectMe (ACCO Server interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Consumer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

Consumer
This parameter contains the identifier of the consumer's LDev in the form of "PDev!LDev".

Parameter type: LPWSTR

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, CBA_S_NOCONNECTION

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_MALFORMED, CBA_E_NOTAPPLICABLE, E_POINTER, RPC_*

7.3.3.4.2.4 SetActivation

The SetActivation service influences the activation state (enable or disable) of one or multiple connections. Table 203 shows the parameters of the service.

Table 203 – SetActivation (ACCO Server interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
State	M	M(=)		
Count	M	M(=)		
List of pProviderID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

State
This parameter contains the State of the connection; i.e. data transmission immediately and/or after the invocation of the SetActivationState service.

Parameter type: VARIANT_BOOL

Count
This parameter contains the number of connections that are to be activated or deactivated.

Parameter type: unsigned long

List of pProviderID
This parameter contains the provider IDs of the connections. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the connection specific faults. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_E_NOTAPPLICABLE, CBA_E_INVALIDID, E_UNEXPECTED

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.3.4.2.5 Ping

The Ping service is invoked by the consumer for heartbeat monitoring if the provider does not send a message to the consumer within a specified dead time. Table 204 shows the parameters of the service.

Table 204 – Ping (ACCO Server interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Consumer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

Consumer
This parameter contains the identifier of the consumer's LDev in the form of "PDev!LDev".

Parameter type: LPWSTR

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServer interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_E_MALFORMED, CBA_E_NOTAPPLICABLE, CBA_S_NOCONNECTION, E_POINTER, RPC_*

7.3.3.4.2.6 Connect2

The Connect service adds one or multiple connection sources to the ACCO object. Table 205 shows the parameters of the service.

NOTE The service request is rejected for the QoSType CONSTANT and is rejected for the QoSType ORPC ST on non system variables.

If a valid reference to a callback interface is supplied with the parameter pICBAAccoCallback, connection data is transferred via push mode; i.e. using the OnDataChanged service. If an invalid reference (NULL) is supplied, connection data is transferred via pull mode; i.e. using the GetConnectionData service.

Table 205 – Connect2 (ACCO Server interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Consumer	M	M(=)		
QoSType	M	M(=)		
QoSValue	M	M(=)		
State	M	M(=)		
pICBAAccoCallback	M	M(=)		
Count	M	M(=)		
List of pConnectIn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pFirstConnect			M	M(=)
List of ppConnectOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServer2 interface of the requested object instance.

Parameter type: Interface Pointer

Consumer
This parameter contains the identifier of the consumer's LDev in the form of "PDev!LDev".

Parameter type: LPWSTR

QoSType
This parameter contains the designation of the QoS type for all connections.

Parameter type: unsigned short

QoSValue
This parameter contains the designation of the QoS value for all connections.

Parameter type: unsigned short

State
This parameter contains the initial state of the connection; i.e. data transmission immediately and/or after the invocation of the SetActivationState service.

Parameter type: VARIANT_BOOL

pICBAAccoCallback
This parameter contains the reference to the callback interface that is used for transferring connection data.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections that are to be connected.

Parameter type: unsigned long

List of pConnectIn
This parameter contains the connection information. The number of elements is determined by the parameter Count.

Parameter type: CONNECTIN2

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServer2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

pFirstConnect
This parameter contains information whether this is the first established connection.

Parameter type: VARIANT_BOOL

List of ppConnectOut
This parameter contains the connection specific results. The number of elements is determined by the parameter Count.

Parameter type: CONNECTOUT

Allowed values for ErrorState (hresult = S_OK): S_OK

Allowed values for ErrorState (hresult = S_FALSE): S_OK, CBA_E_MALFORMED, CBA_E_UNKNOWNOB
CBA_E_COUNTEXCEEDED, CBA_E_SIZEEXCEEDED, E_POINTER, E_ACCESSDENIED, E_UNEXPECT

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServer2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Allowed values: CBA_E_MALFORMED, CBA_E_NOTAPPLICABLE, CBA_E_OUTOFPARTNERACCOS, CBA_E_OUTOFACCOPAIRS, CBA_E_MODECHANGE, E_OUTOFMEMORY, E_FAIL, E_UNEXPECTED, E_INVALIDARG, E_POINTER, RPC_*

7.3.3.4.2.7 GetConnectionData

The GetConnectionData service is invoked by a consumer part of an ACCO object in order to get new connection data via a byte stream of the structure shown in Figure 73.

The maximum size of a GetConnectionData response is device specific. Therefore a subsequent GetConnectionData request may return immediately with pending connection data.

Table 206 shows the parameters of the service.

Table 206 – GetConnectionData (ACCO Server interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Consumer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pLength			M	M(=)
pBuffer			M	M(=)
Header			M	M(=)
Version			M	M(=)
Flags			M	M(=)
Count			M	M(=)
List of Datasets			M	M(=)
ORPC Channel Data Record			S	S(=)
Length			M	M(=)
ConsumerID			M	M(=)
Data			M	M(=)
Data Connection Data			S	S(=)
Quality code			M	M(=)
Value			M	M(=)
RT Channel Data Record			S	S(=)
Length			M	M(=)
Data			M	M(=)
Data Connection Data			S	S(=)
Quality code			M	M(=)
Value			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback interface of the requested object instance.

Parameter type: Interface Pointer

Consumer
This parameter contains the identifier of the consumer's ACCO in the form of „PDev!LDev”.

Parameter type: LPWSTR

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pLength
This parameter contains the overall length of the transfered connection data in octets.

Parameter type: long

pBuffer
This parameter contains the transfered connection data. The data shall be serialized according the used data types with least significant octet first and without any padding octets.

Header
This parameter contains information about the transfered data sets.

Version
This parameter contains the the version of the byte stream.

Parameter type: unsigned char

Allowed Values: according to Table 208

Flags
This parameter describes the structure of the data within the connection data sets. Currently only 0x01 is defined for a .

Parameter type: unsigned char

Allowed Values: according to Table 209

Count
This parameter contains the number of the subsequent data sets that are to be transmitted.

Parameter type: unsigned short

List of Datasets
This parameter contains the list of individual data sets. The number of data sets is determined by the parameter Count in the header.

ORPC Channel Data Record
This parameter contains the data set of a ORPC channel item.

Length
This parameter contains the length of the data set including the length itself.

Parameter type: unsigned short

ConsumerID
This parameter contains identification of the connection sink data in the consumer.

Parameter type: unsigned long

Data
This parameter contains the following subparameters.

Data Connection Data
This parameter contains the new data in the case of a data connection.

Quality code
This parameter contains the Quality code of the data value.

Parameter type: ITEMQUALITYDEF

Value
This parameter contains the value itself with the format according to the appropriate data type.

Parameter type: according to 7.2.2

RT Channel Item Header
This parameter contains the data set of a RT channel item.

Length
This parameter contains the length of the data set including the length itself.

Parameter type: unsigned short

Data
This parameter contains the same subparameters as specified for Data of ORPC Channel Data Record.

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_S_NOCONNECTION, CBA_E_MALFORMED, CBA_E_MODECHANGE, CBA_E_NOTAPPLICABLE, E_OUTOFMEMORY, E_POINTER, RPC_*

The following table contains the allowed data types for Value and Parameter value.

7.3.3.4.3 Interface ACCO Callback

7.3.3.4.3.1 OnDataChanged

The OnDataChanged service is invoked by a provider part of an ACCO object in order to transmit new connection data via a byte stream of the structure shown in Figure 73.

Figure 73 – Structure of the transmitted connection data

Table 207 shows the parameters of the service.

Table 207 – OnDataChanged (ACCO Callback interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Length	M	M(=)		
pBuffer	M	M(=)		
Header	M	M(=)		
Version	M	M(=)		
Flags	M	M(=)		
Count	M	M(=)		
List of Datasets	M	M(=)		
ORPC Channel Data Record	S	S(=)		
Length	M	M(=)		
ConsumerID	M	M(=)		
Data	M	M(=)		
Data Connection Data	S	S(=)		
Quality code	M	M(=)		
Value	M	M(=)		
RT Channel Data Record	S	S(=)		
Length	M	M(=)		
Data	M	M(=)		
Data Connection Data	S	S(=)		
Quality code	M	M(=)		
Value	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback interface of the requested object instance.

Parameter type: Interface Pointer

Length
This parameter contains the overall length of the transfered connection data in octets.

Parameter type: long

pBuffer
This parameter contains the transfered connection data. The data shall be serialized according the used data types with least significant octet first and without any padding octets.

Header
This parameter contains information about the transfered data sets.

Version
This parameter contains the the version of the byte stream.

Parameter type: unsigned char

Allowed Values: according to Table 208

Table 208 – Version

Value	Meaning
0x01	ORPC communication channel
0x11	RT communication channel

Flags
This parameter describes the structure of the data within the connection data sets. Currently only 0x01 is defined for a .

Parameter type: unsigned char

Allowed Values: according to Table 209

Table 209 – Flags

Value	Meaning
0x00	time stamp is not included for each connection data item
0x01	time stamp is included for each connection data item

Count
This parameter contains the number of the subsequent data sets that are to be transmitted.

Parameter type: unsigned short

List of Datasets
This parameter contains the list of individual data sets. The number of data sets is determined by the parameter Count in the header.

ORPC Channel Data Record
This parameter contains the data set of a ORPC channel item.

Length
This parameter contains the length of the data set including the length itself.

Parameter type: unsigned short

ConsumerID
This parameter contains identification of the connection sink data in the consumer.

Parameter type: unsigned long

Data
This parameter contains the following subparameters.

Data Connection Data
This parameter contains the new data in the case of a data connection.

Quality code
This parameter contains the Quality code of the data value.

Parameter type: ITEMQUALITYDEF

Value
This parameter contains the value itself with the format according to the appropriate data type.

Parameter type: according to 7.2.2

RT Channel Item Header
This parameter contains the data set of a RT channel item.

Length
This parameter contains the length of the data set including the length itself.

Parameter type: unsigned short

Data
This parameter contains the same subparameters as specified for Data of ORPC Channel Data Record.

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, RPC_*

7.3.3.4.3.2 Gnip

The Gnip service is invoked by the provider for heartbeat monitoring to allow resource cleanup on communication problems. Gnip stands for Ping reversed, as the Ping service allows the opposite heartbeat monitoring by the consumer.

NOTE The Gnip service should be invoked at least in an interval of 10 seconds. Otherwise the error scenario described in 7.3.3.3.7.4.11 may lead to unexpected behavior.

Table 210 shows the parameters of the service.

Table 210 – Gnip (ACCO Callback interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoCallback2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: RPC_*

7.3.3.4.4 Interface ACCO Sync

7.3.3.4.4.1 ReadItems

The ReadItems service is used for consistent reading of value and quality code. The quality code returned on access to a sub element of a property is always the quality code of the property itself.

NOTE 1 If structures are returned, the value is supplied as a VARIANT stating a SAFEARRAY of unsigned char containing first the type description followed by the content as defined in 7.3.3.4.3.1.

Table 211 shows the parameters of the service.

Table 211 – ReadItems (ACCO Sync interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pReadItem	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppReadItemOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of items that are to be read.

Parameter type: unsigned long

List of pReadItem
This parameter contains the identifiers of the items in the form of "Object.Member" or “Object.MemberAccessList”. The number of elements is determined by the parameter Count.

Parameter type: LPWSTR

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppReadItemOut
This parameter contains the item informations. The number of elements is determined by the parameter Count.

Parameter type: READITEMOUT

Allowed values for ErrorState (hresult = S_OK): S_OK

Allowed values for ErrorState (hresult = S_FALSE): S_OK, CBA_S_VALUEBUFFERED, CBA_S_VALUEUNCERTAIN, CBA_E_NONACCESSIBLE, CBA_E_MALFORMED, CBA_E_UNKNOWNOBJECT, CBA_E_UNKNOWNMEMBER, CBA_E_LIMITVIOLATION, E_FAIL

NOTE 2 CBA_E_LIMITVIOLATION is returned if not initialized data is to be returned, e.g. within a structure.

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.3.4.4.2 WriteItems

The WriteItems service is used for writing data via an identifier.

Writing an item via this service is not allowed, if a connection with a constant is currently assigned to this item or if the item is forced. In both cases the service returns with CBA_E_ACCESSBLOCKED.

Writing an item via this service is allowed, if the item is the consumer of a ORPC or RT connection. How long the value is being kept within an active connection depends on the connection type. For an RT connection the value is overwritten every RT communication cycle; for a ORPC connection the value is overwritten when the next value is transmitted (usally, when the provider has changed).

If sub elements of the requested item are forced, the write will succeed but ignores the values used for the sub elements.

NOTE 1 If a sub element is being written, the user has to be aware that he violates the item's consistency. This depends on the usage model of the item, whether it is more a structuring of multiple independent information or used out of consistency reasons.

NOTE 2 The current quality code of the item is kept, both on writing an item or a sub element.

NOTE 3 If structures are to be written, the value is to be supplied as a VARIANT stating a SAFEARRAY of unsigned char containing first the type description followed by the content as defined in 7.3.3.4.3.1.

Table 212 shows the parameters of the service.

Table 212 – WriteItems (ACCO Sync interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pWriteItem	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of items that are to be written.

Parameter type: unsigned long

pWriteItem
This parameter contains the identifiers of the items in the form of "Object.Member" or "Object.MemberAccessList" and their values. The number of elements is determined by the parameter Count.

Parameter type: WRITEITEMIN

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the item specific errors. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_S_VALUEBUFFERED, CBA_S_VALUEFORCED, CBA_E_MALFORMED, CBA_E_UNKNOWNOBJECT, CBA_E_UNKNOWNMEMBER, CBA_E_TYPEMISMATCH, CBA_E_ACCESSBLOCKED, E_ACCESSDENIED, E_FAIL, RPC_*

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER

7.3.3.4.4.3 WriteItemsQCD

The WriteItemsQCD service is used for consistent writing of value and quality code.

NOTE 1 This service is chiefly used for manually setting a valid value (quasi replacement of a failed connection). The application is the manual transfer of a valid value (e.g. a temperature reading when a communication relationship has failed).

The error code CBA_E_QCNOTAPPLICABLE is returned, if a QualityCode with status “Bad” and any substatus is used. The value will not be set in that case. The same applies if a QualityCode with status “Uncertain” and any substatus but “sensor conversion not accurate” is used for an item of a QC-unaware RT-Auto object.

Writing sub elements via this service is not allowed, i.e. using “Object.MemberAccessList”. In those cases the service returns with CBA_E_MALFORMED.

Writing an item via this service is not allowed, if a connection with a constant is currently assigned to this item or if the item is forced, even in the case where only a sub element is forced. In those cases the service returns with CBA_E_ACCESSBLOCKED.

Writing an item via this service is allowed, if the item is the consumer of a ORPC or RT connection. How long the value is being kept within an active connection depends on the connection type. For an RT connection the value is overwritten every RT communication cycle; for a ORPC connection the value is overwritten when the next value is transmitted (usally, when the provider has changed).

NOTE 2 If structures are to be written, the value is to be supplied as a VARIANT stating a SAFEARRAY of unsigned char containing first the type description followed by the content as defined in 7.3.3.4.3.1.

Table 213 shows the parameters of the service.

Table 213 – WriteItemsQCD (ACCO Sync interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pWriteItemQcdIn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of items that are to be written.

Parameter type: unsigned long

List of pWriteItemQcdIn
This parameter contains the identifiers of the items in the form of "Object.Member" and their values. The number of elements is determined by the parameter Count.

This parameter contains the identifiers of the items and their values. The number of elements is determined by the parameter Count.

Parameter type: WRITEITEMQCDIN

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the item specific errors. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_S_VALUEBUFFERED, CBA_S_VALUEFORCED, CBA_E_MALFORMED, CBA_E_UNKNOWNOBJECT, CBA_E_UNKNOWNMEMBER, CBA_E_TYPEMISMATCH, CBA_E_ACCESSBLOCKED, CBA_E_QCNOTAPPLICABLE, E_ACCESSDENIED, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoSync interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.3.4.5 Interface Group Error

7.3.3.4.5.1 GroupError

The GroupError service reads the error state of an ACCO object. The returned cookie contains whether or not the group error state of the device has changed since the last invocation. The arguments of the service are shown in Table 214.

Table 214 – GroupError (Group Error interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
pMagicCookie			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the group error state of the ACCO object.

Parameter type: GROUPERRORDEF

pMagicCookie
This parameter contains the group error state change indicator.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.3.4.5.2 AdviseGroupError

The AdviseGroupError service requests the server to inform the client if the group error changes.

NOTE 1 The client may wish to be automatically informed about group error changes of an ACCO object. Therefore, it registers itself by means of a AdviceGroupError service delivering an Interface Pointer referencing a client specific GroupErrorEvent interface. This interface is beyond the scope of this standard. However, it is assumed that the client provides an "OnGroupErrorChanged" service with the "NewState" and "OldState" as service parameters that the server calls in case of a group error change.

The arguments of the service are shown in Table 215.

Table 215 – AdviseGroupError (Group Error interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
pGroupErrorEvent	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pCookie			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

pGroupErrorEvent
This parameter contains the address of the event interface.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

NOTE 2 The value S_FALSE is returned if the transferred interface pointer has already been registered as an event interface. The cookie contains the interface pointer of the previous registration. This means, that the interface pointer is not registered again in this case.

pCookie
This parameter contains the event interface handle.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.3.4.5.3 UnadviseGroupError

The UnadviseGroupError service clears the event advise referenced by cookie. The arguments of the service are shown in Table 216.

Table 216 – UnadviseGroupError (Group Error interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Cookie	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

Cookie
This parameter contains the handle to the event interface.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAGroupError interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, CBA_E_INVALIDCOOKIE, RPC_*

7.3.3.4.6 Interface ACCO Server SRT

7.3.3.4.6.1 ConnectCR

The ConnectCR service establishes an AccoDataCR with the ACCO object that acts as the connection provider. The connections transported along with this AccoDataCR will be specified with subsequent invocations of the Connect service. Table 217 shows the parameters of the service.

The parameter pFirstConnect indicates, whether together with processing this request an RT-AR was established. The consumer has to check this against his own expectations. If the consumer awaits an existing RT-AR, while this request returns true or awaits a new RT-AR to be created, while false is returned, the consumer has to invoke the DisconnectMe service and reestablish all connections anew. The error state of all connections regarding this provider will then be set to E_FAIL until reestablished.

If the consumer reconfigures AccoDataCR, he sets the flag CONNECTCR_RECONFIGURE to true. Then the provider uses hidden normally not available resources for the reconfiguration. The consumer is expected to return the resources as fast as possible by disconnecting no longer needed CRs via the DisconnectCR service. The flag Reconfigure shall never be used without promising to return another AccoDataCR. See 7.3.3.3.6.2.5.2 for details.

Table 217 – ConnectCR (ACCO Server SRT interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Consumer	M	M(=)		
QoSType	M	M(=)		
QoSValue	M	M(=)		
pICBAAccoCallback	M	M(=)		
ConsumerMAC	M	M(=)		
Flags	M	M(=)		
Count	M	M(=)		
List of pConnectIn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pFirstConnect			M	M(=)
pProviderMAC			M	M(=)
List of ppConnectOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

Consumer
This parameter contains the identifier of the consumer's LDev in the form of "PDev!LDev".

Parameter type: LPWSTR

QoSType
This parameter contains the RT type for all connections.

Parameter type: unsigned short

QoSValue
This parameter contains the value of RT for all connections.

Parameter type: unsigned short

pICBAAccoCallback
This parameter contains the reference to the callback interface of the consumer to allow gnipping.

Parameter type: Interface Pointer

ConsumerMAC
This parameter contains the MAC address of the consumer.

Parameter type: MACAddr

Flags
This parameter contains flags.

Parameter type: unsigned long

Allowed values: CONNECTCR_RECONFIGURE

Count
This parameter contains the number of CRs that are to be connected.

Parameter type: unsigned long

List of pConnectIn
This parameter contains the CR information. The number of elements is determined by the parameter Count.

Parameter type: CONNECTINCR

NOTE The length stated for an AccoDataCR does not include the frame header.

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

pFirstConnect
This parameter contains information whether the AR_{SRT} was established with this call.

Parameter type: VARIANT_BOOL

pProviderMAC
This parameter contains the MAC address of the provider.

Parameter type: MACAddr

List of ppConnectOut
This parameter contains the CR specific results. The number of elements is determined by the parameter Count.

Parameter type: CONNECTOUTCR

Allowed values for PartialResult(hresult = S_OK): S_OK

Allowed values for PartialResult(hresult = S_FALSE): S_OK, CBA_E_CRDATALENGTH, CBA_E_ COUNTEXCEEDED, CBA_E_FRAMECOUNTUNSUPPORTED, CBA_E_CAPACITYEXCEEDED

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Allowed values: E_POINTER, E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, CBA_E_FLAGUNSUPPORTED, CBA_E_QOSTYPEUNSUPPORTED, CBA_E_QOSTYPENOTAPPLICABLE, CBA_E_QOSVALUEUNSUPPORTED, CBA_E_MALFORMED, CBA_E_OUTOFPARTNERACCOS, CBA_E_OUTOFACCOPAIRS, CBA_E_NOTROUTABLE, CBA_E_ NOTAPPLICABLE, RPC_*

7.3.3.4.6.2 DisconnectCR

The DisconnectCR service removes one or multiple AccoDataCR that were obtained via the ConnectCR service. Provider items transported in the indicated CRs will be automatically disconnected. Table 218 shows the parameters of the service.

Table 218 – DisconnectCR (ACCO Server SRT interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pProviderCRID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of CRs that are to be disconnected.

Parameter type: unsigned long

List of pProviderID
This parameter contains the provider CR IDs of the connections. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the connection specific faults. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_E_INVALIDID, CBA_E_NOTAPPLICABLE

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, RPC_*

7.3.3.4.6.3 Connect

The Connect service establishes a set of connections in a specific AccoDataCR. Table 219 shows the parameters of the service.

Table 219 – Connect (ACCO Server SRT interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
ProviderCRID	M	M(=)		
State	M	M(=)		
LastConnect	M	M(=)		
Count	M	M(=)		
List of pConnectIn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppConnectOut			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

ProviderCRID
This parameter contains the identifier of the AccoDataCR.

Parameter type: unsigned long

State
This parameter contains the initial state of the connection; i.e. data transmission immediately and/or after the invocation of the SetActivationState service.

Parameter type: VARIANT_BOOL

LastConnect
This parameter indicates, whether this is the last connect request to the specified AccoDataCR. The data provided is first send after the call indicated this. If the parameter LastConnect is set to TRUE and no item could be installed, the AccoDataCR is implicitly released and CBA_S_FRAMEEMPTY is returned.

Parameter type: VARIANT_BOOL

Count
This parameter contains the number of items that are to be connected.

Parameter type: unsigned long

List of pConnectIn
This parameter contains the connection information. The number of elements is determined by the parameter Count.

Parameter type: CONNECTINSRT

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppConnectOut
This parameter contains the connection specific results. The number of elements is determined by the parameter Count.

Parameter type: CONNECTOUT

Allowed values for ErrorState (hresult = S_OK): S_OK

Allowed values for ErrorState (hresult = S_FALSE): S_OK, CBA_E_MALFORMED, CBA_E_UNKNOWNOBJECT, CBA_E_UNKNOWNMEMBER, CBA_E_INUSE, CBA_E_ITEMTOOLARGE, CBA_E_TYPEMISMATCH, CBA_E_COUNTEXCEEDED, CBA_E_SIZEEXCEEDED, CBA_E_CRDATALENGTH, E_POINTER, E_ACCESSDENIED, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, CBA_E_INVALIDID, CBA_E_NOTAPPLICABLE, CBA_S_FRAMEEMPTY, RPC_*

7.3.3.4.6.4 Disconnect

The Disconnect service clears one or multiple connections that were obtained via the Connect service. If the last item of a specific AccoDataCR is cleared, the provider releases the AccoDataCR. Table 220 shows the parameters of the service.

Table 220 – Disconnect (ACCO Server SRT interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Count	M	M(=)		
List of pProviderID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

Count
This parameter contains the number of connections that are to be disconnected.

Parameter type: unsigned long

List of pProviderID
This parameter contains the provider IDs of the connections. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the connection specific faults. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_E_INVALIDID, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, RPC_*

7.3.3.4.6.5 DisconnectMe

The DisconnectMe service removes all AccoDataCRs of a consumer. It is typically used by the consumer in its startup in order to clear all existing connections and to reestablish them afterwards. Table 221 shows the parameters of the service.

Table 221 – DisconnectMe (ACCO Server SRT interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Consumer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

Consumer
This parameter contains the identifier of the consumer's ACCO object in the form of "PDev!LDev".

Parameter type: LPWSTR

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, CBA_S_NOCONNECTION

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: CBA_S_NOCONNECTION, CBA_E_MALFORMED, CBA_E_NOTAPPLICABLE, E_POINTER, RPC_*

7.3.3.4.6.6 SetActivation

The SetActivation service modifies the activation state of a set of connections. Table 222 shows the parameters of the service.

Table 222 – SetActivation (ACCO Server SRT interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
State	M	M(=)		
Count	M	M(=)		
List of pProviderID	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
List of ppError			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

State
This parameter contains the activation state of the connection.

Parameter type: VARIANT_BOOL

Count
This parameter contains the number of connections that are to be activated or deactivated.

Parameter type: unsigned long

List of pProviderID
This parameter contains the provider IDs of the connections. The number of elements is determined by the parameter Count.

Parameter type: unsigned long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of ppError
This parameter contains the connection specific faults. The number of elements is determined by the parameter Count.

Parameter type: HRESULT

Allowed values (hresult = S_OK): S_OK

Allowed values (hresult = S_FALSE): S_OK, CBA_E_NOTAPPLICABLE, CBA_E_INVALIDID, E_FAIL

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBAAccoServerSRT interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_FAIL, E_INVALIDARG, E_POINTER, RPC_*

7.3.4 RT-Auto ASE

7.3.4.1 Overview

The runtime automation object (RT-Auto object) ASE represents the automation functionality in the form of a process-related component. Examples are boiler, controller, etc.. These automation functionalities of an application may exist in the form of pre-tested, self-contained and universally useable components. Using the interconnecting functionality that is defined by ACCO ASE, automation functionalities can be configured as a distributed application.

There are two types of RT-Auto objects: System RT-Auto object and Custom RT-Auto objects. The System RT-Auto object offers a System Properties interface for access system information such as operating state and group error whereas the Custom RT-Auto objects offer automation functionality specific interfaces.

To support an easy online layer for an engineering system, an RT-Auto object is not allowed to change its Name or Revision attributes without changing the object instance itself. If an RT-Auto object is to be reconfigured in any of those attributes, the object instance is to be disconnected by using the CoDisconnectObject service and a new object instance is to be created with the new attribute values. The engineering system therefore detects the change through the error code RPC_E_OBJECTDISCONNECTED by using any service on the old disconnected object instance.

7.3.4.2 RT-Auto class specification

7.3.4.2.1 Template

The RT-Auto object is described by the following template:

FAL ASE: **RT-Auto ASE**
CLASS: **RT-Auto**
CLASS ID: **CBA00050-6C97-11D1-8271-00A02442DF7D**
PARENT CLASS: **Base Object**
ATTRIBUTES and SERVICES:

1	(m)	Key Attribute:	Implicite
2	(m)	Attribute:	Interface RT-Auto
2.1	(m)	Attribute:	Name
2.2	(m)	OpsService:	get_Name
2.3	(m)	OpsService:	Revision
2.4	(c)	Constraint:	Base Object Version.Major >= 2
2.4.1	(m)	OpsService:	ComponentInfo
3	(m)	Attribute:	Interface Browse
3.1	(m)	Attribute:	Count
3.2	(m)	OpsService:	get_Count
3.3	(m)	OpsService:	BrowseItems
3.4	(c)	Constraint:	Base Object Version.Major >= 2
3.4.1	(m)	Attribute:	Count2
3.4.2	(m)	OpsService:	get_Count2

3.4.3	(m)	OpsService:	BrowseItems2
5	(m)	Attribute:	Dispatchable

FAL ASE: RT-Auto ASE
CLASS: System RT-Auto
CLASS ID: CBA00090-6C97-11D1-8271-00A02442DF7D
PARENT CLASS: RT-Auto
ATTRIBUTES and SERVICES:

1	(m)	Key Attribute:	Implicite
2	(c)	Constraint:	Base Object Version.Major >= 2
2.1	(m)	Attribute:	Interface System Properties
2.1.1	(m)	Attribute:	StateCollection
2.1.1.1	(m)	Attribute:	LDevState
2.1.1.2	(m)	Attribute:	LDevGroupError
2.1.1.3	(m)	Attribute:	LDevGECookie
2.1.1.4	(m)	Attribute:	AccoGroupError
2.1.1.5	(m)	Attribute:	AccoGECookie
2.1.2	(m)	OpsService:	get_StateCollection
2.1.3	(m)	Attribute:	StampCollection
2.1.3.1	(m)	Attribute:	LDevStamp
2.1.3.2	(m)	Attribute:	AccoStampCDB
2.1.3.3	(m)	Attribute:	AccoStampModCounter
2.1.4	(m)	OpsService:	get_StampCollection

FAL ASE: RT-Auto ASE
CLASS: Custom RT-Auto
CLASS ID: <custom defined>
PARENT CLASS: RT-Auto
ATTRIBUTES and SERVICES:

1	(m)	Key Attribute:	Implicite
2	(m)	Attribute:	List of Custom Interfaces
2.1	(m)	Attribute:	List of Custom Attributes
2.2	(m)	Attribute:	List of Custom Services

7.3.4.2.2 Attributes

Interface RT-Auto
This attribute contains the RT-Auto interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBARTAuto includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBARTAuto2 is derived from the variant ICBARTAuto and includes such the attributes and services of ICBARTAuto and additional attributes and services for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

Name
This attribute contains the name of the RT-Auto object.

Attribute type: BSTR

Interface Browse
This attribute contains the Browse interface of the object. It exists in two variants according to the value of the attribute Base Object Version. The variant named ICBABrowse includes all attributes and services for Base Object Version.Major equal to 1. The variant named ICBABrowse2 is derived from the variant ICBABrowse and includes such the attributes and services of ICBABrowse and additional attributes and services for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

Count
This attribute contains the number of elements in the Browse database.

Attribute type: long

Count2
This attribute contains the number of elements in the Browse database.

Attribute type: long

Dispatchable
This attribute contains the availability of type information for the FAL Class that is provided by the IDispatch common interface. The RT-Auto ASE provides that kind of information and the Dispatchable attribute contains the value 1.

Attribute type: Boolean

Interface System Properties
This attribute contains the System Properties interface of the object and exists in one variant named ICBASystemProperties for Base Object Version.Major greater or equal to 2.

It is a structured attribute composed of the following attributes.

StateCollection
This attribute contains the state collection.

It is a structured attribute composed of the following attributes.

LDevState
This attribute contains the operating state of the LDev object.

Attribute type: STATEDEF

LDevGroupError
This attribute contains the group error of the LDev object.

Attribute type: GROUPERRORDEF

LDevGECookie
This attribute contains the group error cookie of the LDev object.

Attribute type: unsigned short

AccoGroupError
This attribute contains the group error of the ACCO object.

Attribute type: GROUPERRORDEF

AccoGECookie
This attribute contains the group error cookie of the ACCO object.

Attribute type: unsigned short

StampCollection
This attribute contains stamp collection.

It is a structured attribute composed of the following attributes.

LDevStamp
This attribute contains the LDev stamp. If LDev stamp is not supported this attribute contains the value 0.

Attribute type: unsigned long

AccoStampCDB
This attribute contains the ACCO stamp that reflects persistent and volatile changes to the CDB.

Attribute type: unsigned long

AccoStampModCounter
This attribute contains the ACCO stamp modification counter.

Attribute type: unsigned long

List of Custom Interfaces
This attribute contains a list of one or more custom interfaces of the object that shall be defined according to the automation function of a specific RT-Auto Class. The number of custom interfaces is not limited and the actual contents is beyond the scope of this standard. Each element of this structured attribute is composed of the following attributes.

List of Custom Attributes
This attribute contains a list of possible attributes per custom interface. They shall be defined task specific by the manufacturer and are beyond the scope of this standard. These attributes define data that are issue of data connections managed by the ACCO ASE. Subclause 7.2.2 defines the data types that are valid for a data connection.

Only data of the same data type can be connected with each other. For an array, in particular, this means that dimension, length and base data type of source and destination shall be compatible with each other. With a structure, the layout of source and destination shall be compatible (i.e. the data types of the members).

List of Custom Services
This optional attribute contains a list of possible services per custom interface. They shall be defined task specific by the manufacturer and are beyond the scope of this standard.

7.3.4.3 RT-Auto behavior

7.3.4.3.1 General

An RT-Auto object represents the automation functionality as a process-related component. These automation functions exist in the form of pre-tested, self-contained and universally deployable components. Using the interconnecting functionality defined by ACCO ASE, they can be configured as a distributed application.

This functionality, usable for an application, can be subdivided into a fixed and a loadable functionality. A fixed functionality is a function implemented invariably in a LDev. A loadable functionality is programmed in an engineering system and will be loaded into the LDev in a separate step. From the runtime perspective, there is no difference in the outside view onto the components.

7.3.4.3.2 Interconnecting RT-Auto objects

A Custom RT-Auto is characterized by one or multiple interfaces. The manufacturer of the automation object provides these interfaces with defined attributes (properties) and services (methods and events). The following rules are defined with respect to the connection of RT-Auto objects via the ACCO:

– A property for which only one access method with the propget IDL attribute has been defined can be used as the source of a data connection.
– A property for which access methods with the propput and propget IDL attributes have been defined can be used as the sink of a data connection.

Connection types, connection management and productive operations of connections are described in 7.3.3.3

RT-Auto objects shall perform its own first-time initialization (i.e. the initialization during the startup of an RT-Auto object if a connection has not been configured or if a value has not yet been transmitted along a configured connection).

7.3.4.3.3 Property access

This subclause defines the property access behaviour

Full-Access Properties

Accessing a full-access property in the operating state “CBAReady” shall always lead to an acceptance of the value. An RT-Auto object shall provide a buffer for the values that can be

read and written to in the operating state “CBAReady”. All values allotted to the RT-Auto object in this state (either via connection or access to a property) are buffered.

Any access to a full-access property via the RT-Auto interface or the ACCO Sync interface in operating state “CBAReady” shall return CBA_S_VALUEBUFFERED. This hresult value tells the client that the value will not be processed immediately. Since it is a success code, it is of informational purpose, the value returned is valid.

On writing a full-access property, which is currently forced, CBA_S_VALUEFORCED shall be returned to indicate, that the value written is buffered until the force task is released. Reading a forced full-access property shall always return the force value with no further indication, except the quality code being “Good (C) / local override”.

Read-Only Properties

Any access to a read-only property via the RT-Auto interface or the ACCO Sync interface in operating state “CBAReady” shall return CBA_S_VALUEUNCERTAIN. This hresult value tells the client that the state of the value is uncertain. Since it is a success code, it is of informational purpose, the value returned is valid.

System Properties

Access to properties of the System RT-Auto object is independent of the operating state.

Connections

If an RT-Auto object is in state “CBAReady”, the quality code is set „Bad / out of service”. This stimulates the provider to transfer all current values with the new quality code along the connections, since their quality code has changed. Because the inner activity of the RT-Auto objects has stopped (see 7.3.2.2.3.1 for details) this is typically the last transmission during the operating state CBAReady.

The consumer buffers all allotted values at the corresponding RT-Auto objects.

Table 223 defines the hresult values for the access to properties of Custom RT-Auto objects according to the operating state.

Table 223 – Hresult values for access to properties of Custom RT-Auto objects

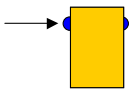
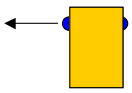
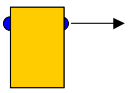
State	Full access property		Read only property
	propput WriteItems 	propget ReadItems 	propget ReadItems 
CBAOperating	S_OK CBA_S_VALUE FORCED	S_OK	S_OK
CBAReady	CBA_S_VALUE BUFFERED	CBA_S_VALUE BUFFERED	CBA_S_VALUE UNCERTAIN
CBANon-Existent	RPC_* (as viewed by the Client)	RPC_* (as viewed by the Client)	RPC_* (as viewed by the Client)

Table 224 defines the hresult values for the access to properties of the System RT-Auto object according to the operating state.

Table 224 – Hresult values for access to properties of the System RT-Auto object

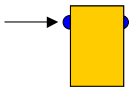
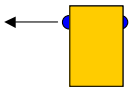
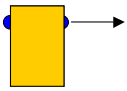
State	Full access property		Read only property
	propput WriteItems 	propget ReadItems 	propget ReadItems 
CBAOperating	S_OK	S_OK	S_OK
CBAReady	S_OK	S_OK	S_OK
CBANon-Existent	RPC_* (as viewed by the Client)	RPC_* (as viewed by the Client)	RPC_* (as viewed by the Client)

Table 225 defines additional hresult values that may be returned independent of the current operating state.

Table 225 – Common hresult values on access to properties of RT-Auto objects

Error code	Description
E_INVALIDARG	The data supported does not match the type of the property.
E_OUTOFMEMORY	Not enough memory.
E_ACCESSDENIED	A write access to a read-only property was attempted.
DISP_E_MEMBERNOTFOUND	The property specified does not exist.
E_UNEXPECTED	Returned on not further specified internal errors.
CBA_E_LIMITVIOLATION	The value is outside the supported range of a specific data type. For a DATE the value exceeds the defined spectrum of DATE (01.01.100 – 31.12.9999). For a BSTR the value exceeds the maximum string size.
CBA_E_TIMEVALUEUNSUPPORTED	The device is not supporting the complete spectrum of DATE (01.01.100 – 31.12.9999) and the date provided falls into that non supported area.

Table 226 defines the quality code for the access to properties of Custom RT-Auto objects according to the operating state.

Table 226 – Quality code for access to properties of Custom RT-Auto objects

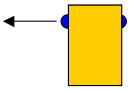
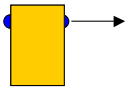
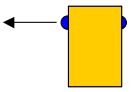
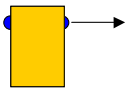
State	Full access property	Read only property
	"HMI connection" ReadItems 	"Productive connection" ReadItems 
CBAOperating	Actual QC (for example of productive connection)	"Good(NC)/ok", resp. according to the rules of the provider
CBAReady	"Bad / out of service" On transition to CBAReady transmitted along connections	"Bad / out of service" On transition to CBAReady transmitted along connections
CBANon-Existent	Consumer delivers QC regarding to substitute value strategy	Consumer delivers QC regarding to substitute value strategy

Table 227 defines the quality code for the access to properties of the System RT-Auto object according to the operating state.

Table 227 – Quality code for access to properties of the System RT-Auto object

State	Full access property	Read only property
	"HMI connection" ReadItems 	"Productive connection" ReadItems 
CBAOperating	"Good(NC)/ok"	"Good(NC)/ok"
CBAReady	"Good(NC)/ok"	"Good(NC)/ok"
CBANon-Existent	Consumer delivers QC regarding to substitute value strategy	Consumer delivers QC regarding to substitute value strategy

The Quality code is transmitted along the connection by the provider of the connections (except for the state CBANonExistent, where it is built implicitly by the consumer), and also returned by access to the property via the ReadItems service of the ACCO Sync interface.

7.3.4.4 RT-Auto service specification

7.3.4.4.1 Interface RT-Auto

7.3.4.4.1.1 get_Name

The get_Name service reads the name of an RT-Auto object. The name is used for navigation. The arguments of the service are shown in Table 228. The service returns the attribute Name.

Table 228 – get_Name (RT-Auto interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBARTAuto interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBARTAuto interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the name of the RT-Auto object.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBARTAuto interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.4.4.1.2 Revision

The Revision service reads the revision level of the RT-Auto (i.e. the software version) as a major and minor revision. The arguments of the service are shown in Table 229.

Table 229 – Revision (RT-Auto interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pMajor			M	M(=)
pMinor			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBARTAuto interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBARTAuto interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pMajor
This parameter contains the major part of the revision.

Parameter type: short

pMinor
This parameter contains the minor part of the revision.

Parameter type: short

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBARTAuto interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.4.4.1.3 ComponentInfo

The ComponentInfo service reads the component information of an RT-Auto object.

NOTE The ComponentID reflects the engineering component's ID – the one of the component's description.

The arguments of the service are shown in Table 230.

Table 230 – ComponentInfo (RT-Auto interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pComponentID			M	M(=)
pVersion			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBARTAuto2 interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBARTAuto2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pComponentID
This parameter contains the component identifier. For a System RT-Auto object the NULL-GUID will be returned.

Parameter type: BSTR

pVersion
This parameter contains the version. For a Ssytem RT-Auto object a NULL-Version will be returned.

Parameter type: BSTR

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBARTAuto2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.4.4.2 Interface Browse

7.3.4.4.2.1 get_Count

The get_Count service reads the number of elements in the Browse database. The arguments of the service are shown inTable 231. This service returns the attribute Count.

NOTE Non-dispatch clients should use the get_Count2 service.

Table 231 – get_Count (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
pVal	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the number of elements.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.4.4.2.2 BrowseItems

The BrowseItems service reads the list of available properties of an RT-Auto object. The arguments of the service are shown in Table 232.

The server is able to control the maximum amount of returned information via the number of items in the SAFEARRAY. If it obtains less than the number of items indicated via the get_Count service, the client shall continue invoking the service for the subsequent items. To do this, it shall advance the Offset parameter accordingly (by adding the number of returned items, beginning with 0).

The client, in turn, can use the MaxReturn parameter to control the maximum number of returned items. The server shall not return more than this number of items. If the client sets this parameter to 0, the server still returns S_OK, but no items in the SAFEARRAY.

NOTE 1 The client should be able to cope with the fact that the object world has changed between invocations of the get_Count and BrowseItems services. A distributed automation device does not maintain any status information.

NOTE 2 Non-dispatch clients should use the BrowseItems2 service to get extended informations.

Table 232 – BrowseItems (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Offset	M	M(=)		
MaxReturn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pItem			M	M(=)
pDataType			M	M(=)
pAccessRight			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

Offset
This parameter contains the offset of the first requested RT-Auto item.

Parameter type: long

MaxReturn
This parameter contains the maximum number of received items.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pItem
This parameter contains the name or list of names of the available properties.

Parameter type: VARIANT

pDataType
This parameter contains data type or list of data types of the available properties.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_I4, Value according to VARTYPE

pAccessRight
This parameter contains the access privileges or list of access privileges of the available properties.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_I4, Value according ACCESSRIGHTSDEF

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, RPC_*

7.3.4.4.2.3 get_Count2

The get_Count2 service reads the number of elements in the Browse database according to the specified selector. The arguments of the service are shown in Table 233. This service reads the attribute Count.

Table 233 – get_Count2 (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Selector	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

Selector
This parameter contains the items to count. Selector "0" is used for reading the number of properties that are related to the referred RT-Auto object.

Parameter type: long

Allowed values: 0

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the number of Browse items.

Parameter type: long

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, CBA_E_INVALIDENUMVALUE, RPC_*

7.3.4.4.2.4 BrowseItems2

The BrowseItems2 service reads the list of available items of an RT-Auto object according to the specified selector. The arguments of the service are shown in Table 234.

The server is able to control the maximum amount of returned information via the number of items in the SAFEARRAY. If it obtains less than the number of items indicated via the get_Count service, the client shall continue invoking the service for the subsequent items. To do this, it shall advance the Offset parameter accordingly (by adding the number of returned items, beginning with 0).

The client, in turn, can use the MaxReturn parameter to control the maximum number of returned items. The server shall not return more than this number of items. If the client sets this parameter to 0, the server still returns S_OK, but no items in the SAFEARRAY.

NOTE The client should be able to cope with the fact that the object world has changed between invovations of the get_Count and BrowseItems services. A distributed automation device does not maintain any status information.

Table 234 – BrowseItems2 (Browse interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Selector	M	M(=)		
Offset	M	M(=)		
MaxReturn	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pItem			M	M(=)
pInfo1			U	U(=)
pInfo2			U	U(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

Selector
This parameter contains the items to browse. Selector “0” is used for reading a list of properties ... that are related to the referred RT-Auto object.

Parameter type: long

Allowed values: 0

Offset
This parameter contains the offset of the first item that shall be returned.

Parameter type: long

MaxReturn
This parameter contains the maximum number of items that shall be returned.

Parameter type: long

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pItem
This parameter contains the name or list of names of the available items that correspondes to the parameter selector.

For selector “0” it contains the list of properties of the RT-Auto object.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_BSTR

pInfo1
This parameter contains information corresponding to the parameter selector.

For selector “0” it contains the extended type descriptions of the returned items as defined in 7.2.2.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_BSTR

pInfo2
This parameter contains information corresponding to the parameter selector.

For selector “0” it contains the access rights of the returned items.

Parameter type: VARIANT

Allowed values: Tag=VT_SAFEARRAY_I4, Value according to ACCESSRIGHTSDEF.

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBABrowse2 interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, CBA_E_INVALIDENUMVALUE, RPC

7.3.4.4.3 Interface system properties

7.3.4.4.3.1 get_StateCollection

The get_StateCollection service reads the state collection of a System RT-Auto object. The state collection contains a quick overview over the operating state and group error of the LDev object as well as the group error of the ACCO object. The arguments of the service are shown in Table 235. The service returns the attribute StateCollection of the System Properties interface.

Table 235 – get_StateCollection (System properties interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBASystemProperties interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBASystemProperties interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the attribute StateCollection.

Parameter type: SAFEARRAY (unsigned char)

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBASystemProperties interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.3.4.4.3.2 get_StampCollection

The get_StateCollection service reads the stamp collection of a System RT-Auto object. The stamp collection contains a quick overview over the LDev stamp as well as the ACCO stamps (see 7.1.7 for details). The arguments of the service are shown in Table 236. The service returns the attribute StampCollection of the System Properties interface.

Table 236 – get_StampCollection (System Properties interface)

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)
pVal			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the ICBASystemProperties interface of the requested object instance.

Parameter type: Interface Pointer

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the ICBASystemProperties interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

pVal
This parameter contains the attribute StampCollection.

Parameter type: SAFEARRAY (unsigned char)

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the ICBASystemProperties interface of the requested object instance.

Parameter type: Interface Pointer

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: E_INVALIDARG, E_OUTOFMEMORY, E_POINTER, RPC_*

7.4 ARs

7.4.1 Overview

The FAL for distributed automation provides exactly one AREP class referred to as ORPC AR. It provides many-to-many, bi-directional, user-triggered and queued application relation utilizing the client server model in both roles. Each device shall implement exactly one instance of that class to support the transmission of all distributed automation ASE services by means of the AR-Call service. Therefore, it multiplexes and de-multiplexes all communication paths. The path or binding information derived from the common service

parameter Interface Pointer is maintained by the abstract ORPC instance as the lower part of the application layer. By means of the AR-CoCreateInstance service an application process gets this binding information to a remote object instance with the appropriate Interface Pointer within the response. The host and socket information is stored by the abstract ORPC sublayer and is no subject of distributed automation AR or APO ASEs. The distributed automation object model only allows the usage of the AR-CoCreateInstance service for the Physical Device class (PDev). Following the distributed automation navigation procedure it is possible to get the binding information to other distributed automation ASE objects (LDev, ACCO, RT-Auto).

7.4.2 ORPC AR class specification

7.4.2.1 Formal definition

FAL ASE: **AR ASE**
CLASS: **ORPC AR**
CLASS ID: **Not Used**
PARENT CLASS: **TOP**
ATTRIBUTES and SERVICES:

1	(m)	Attribute:	IDL Marshaling Reference
2	(m)	Attribute:	ORPC Mapping Reference
3	(m)	Attribute:	ORPC Binding Reference
4	(m)	OpsService:	CoCreateInstance
5	(m)	OpsService:	CoDisconnectObject
6	(m)	OpsService:	Call

7.4.2.2 Attributes

IDL Marshaling Reference
This attribute contains the usage of the interface definition language (IDL) to marshal the service parameter into the ORPC-PDUs as defined in this standard.

ORPC Mapping Reference
This attribute contains the mapping of the AR services into the ORPC services of the abstract ORPC model as defined in IEC/TR 61158–1:2010.

ORPC Binding Reference
This attribute contains the binding of the Interface Pointer of the AR services into the additional host and socket information that are maintained by the abstract ORPC model. The used abstract ORPC model provides the management of that information for the convenience of APO ASEs and AR ASEs.

7.4.3 ORPC AR service specification

7.4.3.1 CoCreateInstance

The CoCreateInstance service creates a binding to a remote instance of a FAL ASE. The arguments of the service are shown in Table 237.

Table 237 – CoCreateInstance

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Host address	M	M(=)		
Class ID	M	M(=)		
Interface ID	M	M(=)		
Result(+)			S	S(=)
hresult			M	M(=)
Interface Pointer			M	M(=)
Result(-)			S	S(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Host Address
This parameter contains the address information of the remote node as IP address.

Parameter type: VARIANT

Class ID
This parameter contains the unique class identifier of the remote object. Only the Class ID of the Physical Device Class is valid.

Parameter type: UUID

Allowed Values: UUID_PhysicalDevice

Interface ID
This parameter contains the unique interface identifier of the requested interface of the remote object.

Parameter type: UUID

Result(+):
This selection type parameter indicates that the service request succeeded.

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

Interface Pointer
This parameter contains the address of the requested interface of the remote object.

Parameter type: Interface Pointer

Result(-):
This selection type parameter indicates that the service request failed.

hresult
This parameter contains the result code.

Parameter type: HRESULT

7.4.3.2 CoDisconnectObject

The CoDisconnectObject service enables a server to correctly disconnect all external clients to the object specified by the interface pointer. The arguments of the service are shown in Table 238.

Table 238 – CoDisconnectObject

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Result			M	M(=)
Interface Pointer			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of an interface of the object to be disconnected.

Parameter type: Interface Pointer

Result:
This selection type parameter indicates that the service request succeeded.

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK

7.4.3.3 Call

The Call service transports all distributed automation ASE services. The arguments of the service are shown in Table 239.

Table 239 – Call

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Interface Pointer	M	M(=)		
Service Name	M	M(=)		
List of Unified Service In-Parameter	M	M(=)		
Result(+)			S	S(=)
Interface Pointer			M	M(=)
Service Name			M	M(=)
hresult			M	M(=)
List of Unified Service Out-Parameter			M	M(=)
Result(-)			S	S(=)
Interface Pointer			M	M(=)
Service Name			M	M(=)
hresult			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

Interface Pointer
This parameter contains the address of the requested interface.

Parameter type: Interface Pointer

Service Name
This parameter contains the name or method number of the distributed automation ASE service that is used to convey the application data.

Parameter type: VARIANT

List of Unified Service In-Parameter
This parameter contains all parameters that are issued by the distributed automation ASE service. By means of the IDL information and the service and interface name the abstract ORPC model provides the marshaling of the service parameter and codes the ORPC-PDU.

Parameter type: VARIANT

Result(+):
This selection type parameter indicates that the service request succeeded.

Interface Pointer
This parameter contains the address of the requested interface.

Parameter type: Interface Pointer

Service Name
This parameter contains the name or method number of the distributed automation ASE service that is used to convey the application data.

Parameter type: VARIANT

hresult
This parameter contains the result code.

Parameter type: HRESULT

Allowed values: S_OK, S_FALSE

List of Unified Service Out-Parameter
This parameter contains all parameters that are issued by the distributed automation ASE service. This parameter is interpreted according to the parameter definition of the called distributed automation ASE service.

Parameter type: VARIANT

Result(-):
This selection type parameter indicates that the service request failed.

Interface Pointer
This parameter contains the address of the requested interface.

Parameter type: Interface Pointer

Service Name
This parameter contains the name or method number of the distributed automation ASE service that is used to convey the application data.

Parameter type: VARIANT

hresult
This parameter contains the result code.

Parameter type: HRESULT

7.5 Summary of FAL classes

This subclause contains a summary of the defined fieldbus AL classes.

Table 240 shows the fieldbus AL class summary.

Table 240 – Distributed automation FAL class summary

AL ASE	Class
AP	Physical device Logical Device
ACCO	ACCO
RT-Auto	RT-Auto System RT-Auto Custom RT-Auto

7.6 Summary of FAL services

Table 241 gives an overview of the assignment of services to the client and the server.

Table 241 – Assignment of the services to client and server

Service	Client		Server	
	Req	Cnf	Ind	Rsp
Physical Device				
Interface Unknown				
QueryInterface	x	x	x	x
AddRef	x	x	x	x
Release	x	x	x	x
Interface Dispatch				
GetTypeInfoCount	x	x	x	x
GetTypeInfo	x	x	x	x
GetIDsOfNames	x	x	x	x
Invoke	x	x	x	x
Interface Physical Device				
get_Producer	x	x	x	x
get_Product	x	x	x	x
get_SerialNo	x	x	x	x
get_ProductionDate	x	x	x	x
Revision	x	x	x	x
get_LogicalDevice	x	x	x	x
type	x	x	x	x
PROFInetRevision	x	x	x	x
get_PDevStamp	x	x	x	x
Interface Browse				
get_Count	x	x	x	x
BrowseItems	x	x	x	x
get_Count2	x	x	x	x
BrowseItems2	x	x	x	x
Interface Persist				
Save	x	x	x	x
Save2	x	x	x	x
Logical Device				
Interface Unknown				
QueryInterface	x	x	x	x
AddRef	x	x	x	x
Release	x	x	x	x
Interface Dispatch				
GetTypeInfoCount	x	x	x	x
GetTypeInfo	x	x	x	x
GetIDsOfNames	x	x	x	x
Invoke	x	x	x	x
Interface Logical Device				
get_Name	x	x	x	x
get_Producer	x	x	x	x
get_Product	x	x	x	x
get_SerialNo	x	x	x	x
get_ProductionDate	x	x	x	x
Revision	x	x	x	x
get_ACCO	x	x	x	x
get_RTAuto	x	x	x	x
PROFInetRevision	x	x	x	x
ComponentInfo	x	x	x	x
Interface State				
get_State	x	x	x	x
Activate	x	x	x	x
Deactivate	x	x	x	x
Reset	x	x	x	x
AdviseState	x	x	x	x

Service	Client		Server	
	Req	Cnf	Ind	Rsp
UnadviseState	x	x	x	x
Interface Time				
get_Time	x	x	x	x
put_Time	x	x	x	x
Interface Browse				
get_Count	x	x	x	x
BrowseItems	x	x	x	x
get_Count2	x	x	x	x
BrowseItems2	x	x	x	x
Interface Group Error				
GroupError	x	x	x	x
AdviseGroupError	x	x	x	x
UnadviseGroupError	x	x	x	x
ACCO				
Interface Unknown				
QueryInterface	x	x	x	x
AddRef	x	x	x	x
Release	x	x	x	x
Interface Dispatch				
GetTypeInfoCount	x	x	x	x
GetTypeInfo	x	x	x	x
GetIDsOfNames	x	x	x	x
Invoke	x	x	x	x
Interface ACCO Management				
AddConnections	x	x	x	x
RemoveConnections	x	x	x	x
ClearConnections	x	x	x	x
SetActivationState	x	x	x	x
GetInfo	x	x	x	x
GetIDs	x	x	x	x
GetConnections	x	x	x	x
ReviseQoS	x	x	x	x
get_PingFactor	x	x	x	x
put_PingFactor	x	x	x	x
get_CDBCookie	x	x	x	x
GetConsIDs	x	x	x	x
GetConsConnections	x	x	x	x
DiagConsConnections	x	x	x	x
GetProvIDs	x	x	x	x
GetProvConnections	x	x	x	x
GetDiagnosis	x	x	x	x
Interface ACCO Server				
Connect	x	x	x	x
Disconnect	x	x	x	x
DisconnectMe	x	x	x	x
SetActivation	x	x	x	x
Ping	x	x	x	x
Connect2	x	x	x	x
GetConnectionData	x	x	x	x
Interface ACCO Callback				
OnDataChanged	x	x	x	x
Gnip	x	x	x	x
Interface ACCO Sync				
ReadItems	x	x	x	x
WriteItems	x	x	x	x
WriteItemsQCD	x	x	x	x
Interface Group Error				
GroupError	x	x	x	x
AdviseGroupError	x	x	x	x
UnadviseGroupError	x	x	x	x

Service	Client		Server	
	Req	Cnf	Ind	Rsp
Interface ACCO Server SRT				
ConnectCR	x	x	x	x
DisconnectCR	x	x	x	x
Connect	x	x	x	x
Disconnect	x	x	x	x
DisconnectMe	x	x	x	x
SetActivation	x	x	x	x
RT-Auto				
Interface Unknown				
QueryInterface	x	x	x	x
AddRef	x	x	x	x
Release	x	x	x	x
Interface Dispatch				
GetTypeInfoCount	x	x	x	x
GetTypeInfo	x	x	x	x
GetIDsOfNames	x	x	x	x
Invoke	x	x	x	x
Interface RT-Auto				
get_Name	x	x	x	x
Revision	x	x	x	x
Interface Browse				
get_Count	x	x	x	x
BrowseItems	x	x	x	x
get_Count2	x	x	x	x
BrowseItems2	x	x	x	x
Interface System Properties				
get_StateCollection	x	x	x	x
get_StampCollection	x	x	x	x
ORPC AR				
CoCreateInstance	x	x	x	x
CoDisconnectObject	x	x	x	x
Call	x	x	x	x

8 Communication model for decentralized periphery

8.1 Concepts

8.1.1 User requirements

The typical automation system consists of one or several programmable controllers connected with IO-systems to the machine/process. Today, an IO-system is mostly implemented as a hierarchical decentralized system or a set of single point connections with a well defined electrical interface such as 24V digital signals or 4 to 20mA analogue signals.

This communication model defines a decentralized system and the connection is based on ISO/IEC 8802-3 information and telecommunication technology. The defined object model is an enhancement of the IEC 61158–5–3 object model.

8.1.2 Features

Combining the structure defined in IEC 61131–1 the following advantages are achieved:

- Changes in the plant cause only small changes in wiring
- Critical signals can be transferred over short distances to the remote field device (sensor/actuator) and to/from the programmable controller over long distances with high accuracy
- Parameterization and diagnosis can be done without additional wiring
- Commissioning of subparts without additional installations

Table 242 gives an overview of the requirements and features of the system.

Table 242 – Requirements and features

Requirement	Feature
Short reaction time	Cyclic exchange of more than 1 000 inputs and outputs with 32 field devices in less than 1 ms
Automation system consists of one or several programmable controller	Mono-controller or Multi-controller operation, more than one IO controller can be associated to one IO device
Simple field device	Simple protocol, low cost communication interface
Adaptation of application model of IEC 61158–5–3 / IEC 61158–6–3	Similar application process object model with compatibility modus
Unified protocols	Use of the same real-time transport protocol mechanism for different application models
Excellent diagnosis	Diagnosis and alarm object model for IO devices
Smart field devices	Acyclic communication which provides a flexible and enhanced addressing scheme of data within IO devices. The possibility to transfer an alarm from the IO device to the IO controller and vice versa with an explicit acknowledgement.
Efficient communication between IO devices and between IO controller	Provider/Consumer mechanism
Synchronization of applications	Isochronous mode, jitter less than 1 µs, signaling to the application
Synchronization of clocks	High precision time synchronization
Redundancy	IO device and IO controller redundancy, support for media redundancy
Interoperability	Precise and complete definitions including the definition of the system behavior
Modification during operation	Dynamic reconfiguration

8.1.3 Associations

The communication model supports three types of associations which are characterized by device types involved in the association.

NOTE 1 Additional associations e.g. for upload or download may be possible but are not part of the communication model.

The communication model supports the communication of field devices with one ore more controlling devices (e.g. programmable controller or distributed control system) via associations (see Figure 74). It includes cyclic data exchange of IO data and transmission of alarms, transmission of parameter data, configuration data, identification data, logbook data, diagnosis data, and record data.

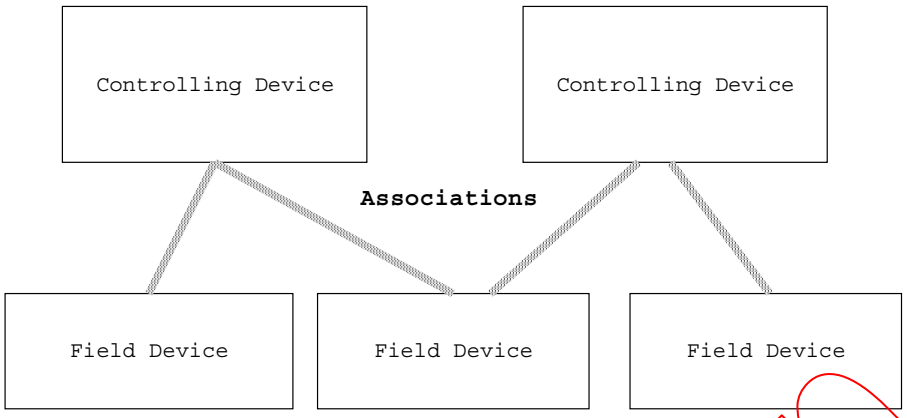


Figure 74 – Example of communication between controlling devices and field devices

Furthermore, the communication between an engineering device and several controlling or field devices is also supported (see Figure 75). The association between engineering and field devices includes transmission of parameter data, configuration data, identification data, logbook data, diagnosis data, and record data via connectionless RPC protocol.

NOTE 2 The association between engineering and controlling device includes transmission of controller diagnosis, domains and commands for device activation via connectionless RPC protocol (for further studies).

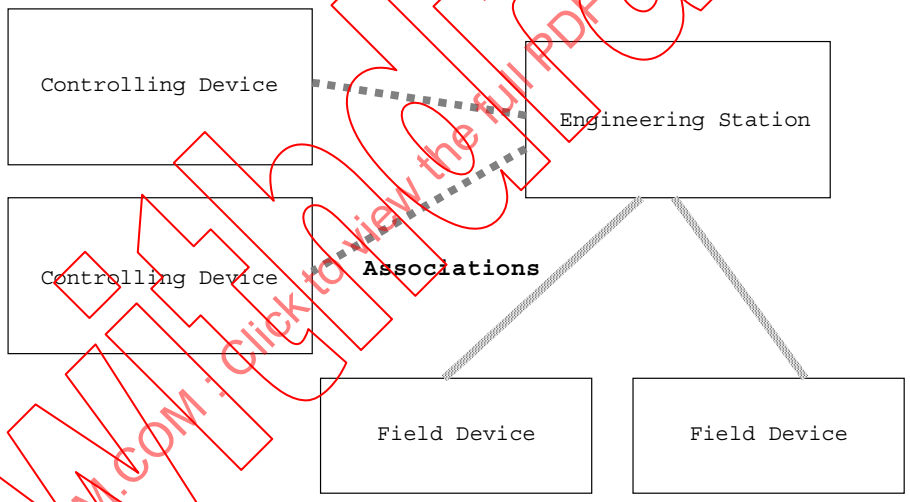


Figure 75 – Example of communication between an engineering station and several controlling and field devices

In addition to these communication models, it also supports the communication between a server station and field devices (see Figure 76). The association between a server station and field device includes transmission of record data via connectionless RPC protocol.

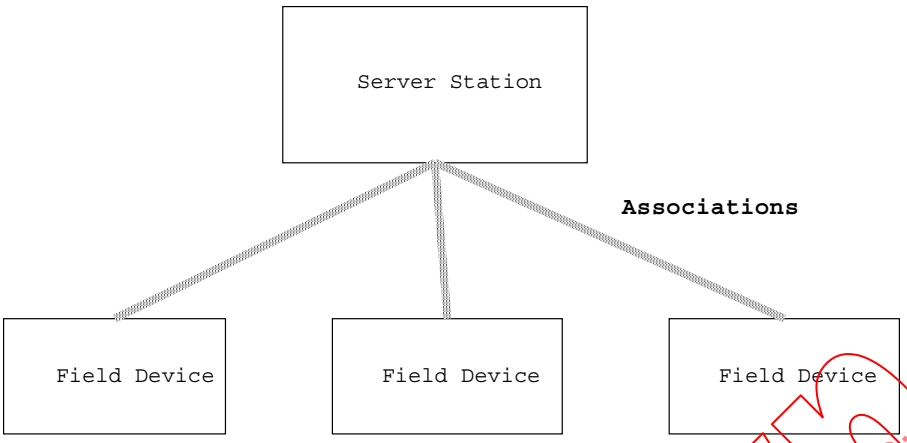


Figure 76 – Example of communication between field devices and a server station

In addition to these communication models, it also supports the communication between field devices (see Figure 77). This communication model includes a controlling device to setup and control the association between the field devices.

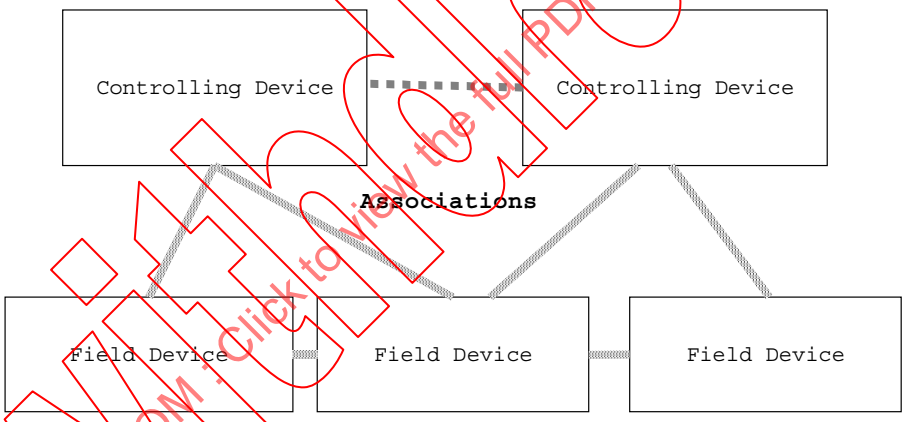


Figure 77 – Example of communication between field devices

NOTE 3 The association between controlling devices may include e.g. the transmission of IO data, diagnosis, alarms and parameter

8.1.4 Device types

8.1.4.1 General

A physical automation device may host one ore more of the same or different device types with appropriate associations.

8.1.4.2 IO controller

The IO controller is a controlling device which is associated with one or more IO devices (field devices). The IO controller performs one or more of the following functionalities:

a) cyclic functionalities
   - exchange of IO data with related IO devices

b) acyclic functionalities
- read diagnosis from IO devices
- configuration of IO devices
- writes parameter data to IO devices (startup or application parameter)
- treatment of configuration and diagnosis requests of an engineering device
- establish context to IO devices by means of context management
- acyclic access to record data of IO devices
- treatment of alarms from IO devices
- sending of alarms to IO devices

c) general functionalities
- redundancy
- dynamic reconfiguration
- isochronous operation
- provider/consumer communication between IO devices and IO controllers
- clock synchronization

8.1.4.3 IO supervisor

The IO supervisor is an engineering device which manages provision of configuration data (parameter sets) and collection of diagnosis data for/from a IO controllers and/or IO devices.

8.1.4.4 IO parameter server

The IO parameter server is a server station which serves to store and to load application configuration data (record data objects) from IO devices (client).

8.1.4.5 IO device

8.1.4.5.1 General

The IO device is a field device and performs the following activities depending on the functionality.

The functionalities are:

a) cyclic functionalities
- exchange of IO data with one ore more assigned IO controller
- exchange of IO data with related IO devices

b) acyclic functionalities
- provisions for diagnosis data supply
- treatment of configuration requests of an IO controller
- provision of acyclic access to record data for an IO controller
- provision of alarms to the assigned IO controller
- treatment of parameters (startup or application parameter)
- treatment of configuration and diagnosis requests of an engineering device
- treatment of alarms from IO controllers
- sending of alarms to IO controllers

c) general functionalities

- provider/consumer communication between IO devices and IO controllers
- redundancy
- dynamic reconfiguration
- isochronous operation
- clock synchronization

IO device conformance classes are defined to support common functions.

In general, an IO device is hierarchical composed including the following elements:

- one or more IO device instances (see Annex A)
- each IO device instance includes one or more application processes referenced by its identifier (API)
- each API includes one or more slots
- each slot includes one or more subslots
- each subslots may include one or more channels

The IO device is composed of different structural units to group application process objects and provides a certain level of abstraction. These structural units may reflect hardware components or virtual functional units of the field device. The aim is to provide a suitable set of address parameter. These structural units are referred to as slots, subslots, and channels which may reflect also physical units, sub-units, or a single connection point of an IO device. Figure 78 shows the structure for one arbitrary API. The structural units instance, api, slot and subslot are accessed by means of Object UUID, API Number, Slot Number and Subslot Number within the address model.

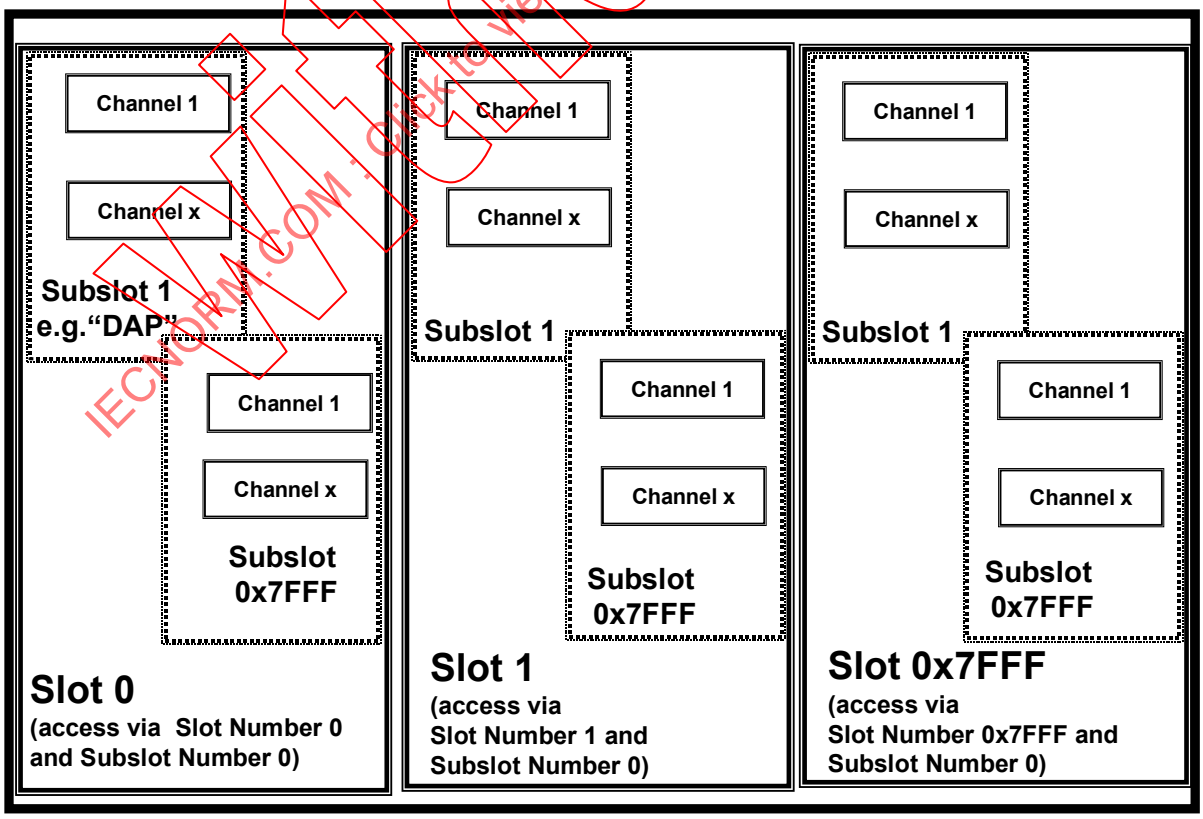


Figure 78 – Structural units of one arbitrary API of an IO device (general)

Furthermore, subslots 0 to 0x7FFF may be used in all APIs of an IO device in parallel. A particular application relationship may be established to more than one application processes. An IO device instance with more than one Application Process Identifier referred to as Multi-API-Instance.

NOTE An IO device instance supporting a particular profile API is a Multi-API-Instance because the API 0 support of the IO device instance is always mandatory.

A device manufacturer may define a particular slot/subslot combination to represent the IO device itself. It is referred to as Device Access Point (DAP). The DAP submodule shall be described within the GSDML.

The Subslot Number 0 shall be used to represent the module in the special case “pull module”, if only the “pull” alarm is supported. In this case, a “real” Subslot 0 does not exist and shall not contain IO channels, diagnosis or record data. If the “pull module” alarm is supported, a “real” Subslot 0 exist, and Subslot 0 may contain IO channels, diagnosis or record data.

Additionally, special subslots in the range from 0x8000 to 0x8FFF are defined for API 0. An IO device shall contain at least one interface with ports. Therefore, at least one Slot (e.g. Slot Number 0), shall contain up to 16 further special Subslots from 0x8000 (Interface 1), 0x8100 (Interface 2), to 0x8F00 (Interface 16) referred to as Interface Subslots. Interface Subslots define the remote access to IP address parameters and the name of the interface (DNS name). Furthermore, each Interface Subslot shall have at least one and up to 255 assigned ports. These subslots from 0x8x01(Port 1) to 0x8xFF (Port 255) referred to as Port Subslots. The subslot defines the remote access to port specific parameter e.g. Own Port ID. Figure 79 depicts an example with several ports on the same slot belonging to one interface. All these special subslots shall only be used within API 0. The values “port-001” to “port-255” shall be used for naming the port submodules if all port submodules belonging to this interface are located in the same slot as the interface submodule.

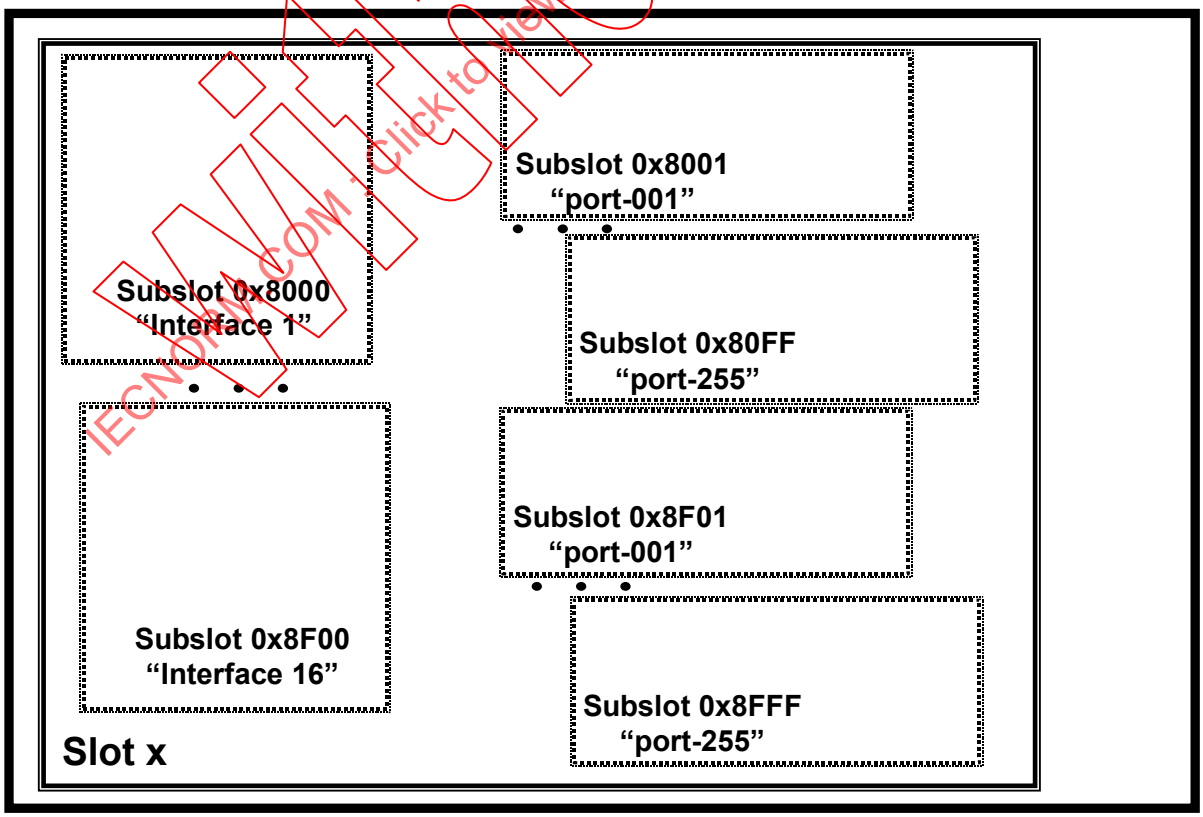


Figure 79 – Example 1 structural units for interfaces and ports within API 0

Figure 80 depicts an example where several port submodules belonging to an interface are located on several slots. The values “port-001-xxxxx” to “port-255-zzzzz” shall then be used for naming the port submodules, whereas “xxxxx” denoted the slot number with leading “0” (e.g. port-002-00005”).

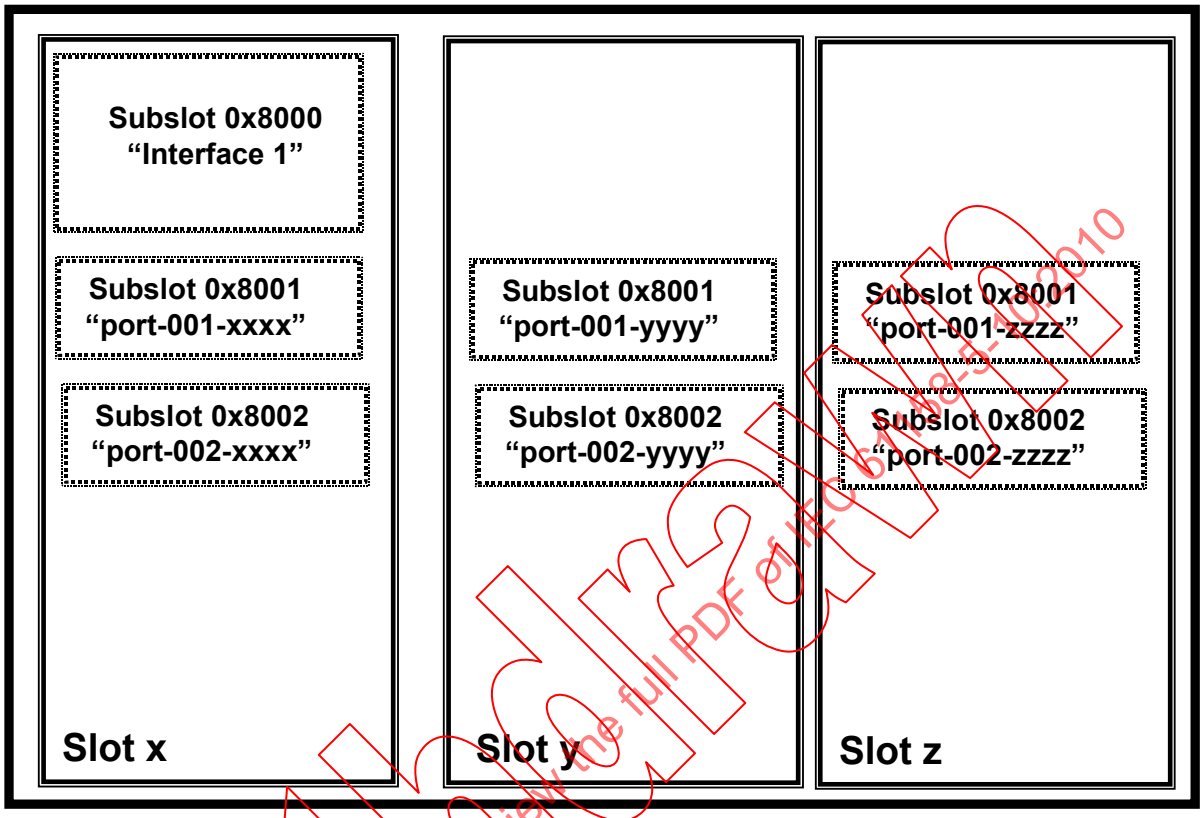


Figure 80 – Example 2 structural units for interfaces and ports within API 0

All possible configurations of an IO device should be described in the GSDML file of the device.

8.1.4.5.2 Slot

The application layer uses slots to reflect the structuring of functions or components (e.g. hardware modules, logical units) inside an IO device. A slot shall have one or more subslots which represent the further level to structure the data.

Each slot shall be addressed by a slot number starting with one. The last possible Slot Number is 0x7FFF. Numbering may contain gaps.

NOTE The slot model is an abstract address model via numeric references. Real hardware modules or virtual functional units may even occupy more than one slot. The device characteristics concerning use of slots are defined in a general station description for configuration purpose.

Furthermore, each module owns configuration data. Within the connection establishment phase the requested configuration shall be compared with the real configuration data. If a slot is not occupied with a physical or virtual module then it is not part of the configuration.

In modular IO devices slots determine the actual configuration of the field device. The number of slots is fixed according to the device specification.

A compact IO device is a special case of a modular IO device with one or more fixed configurations.

8.1.4.5.3 Subslot

The application layer uses subslots to reflect the structuring of functions or further components (e.g. hardware units, logical units) inside a slot of an IO device. A subslot may have one or more channels which represent the real structure of the Input and/or Output Data. These channels may be a further subdivision of the Input and/or Output Data object.

Each subslot will be addressed by a subslot number 0 to 0x7FFF. Furthermore, within a slot special subslots 0x8000 to 0x8FFF for interfaces and related ports may exist.

If ARProperties.PullModuleAlarmAllowed is not used, subslot number 0 is used in conjunction with AlarmType “Pull” to address a certain module within the IO device. In this case, it shall not contain channels and IO data.

If ARProperties.PullModuleAlarmAllowed is used, subslot number 0 acts as a real subslot. On the other hand, real subslots may contain channels and IO data. The actual use of subslot number and index to address data within the device is manufacturer specific.

Subslot 0 is addressing the module within the scope of the Pull Alarm to signal pulling of a module, if ARProperties.PullModuleAlarmAllowed is not used. If ARProperties.PullModuleAlarmAllowed is used, Subslot 1 to 0x7FFF is freely usable within the scope of the addressed AP.

Within a slot, a subslot provides one or several of the following objects:

- Record Data (data with open semantics, e.g. parameter), addressed by
  - slot number (0 to 0x7FFF),
  - subslot number (0 to 0x8FFF or else 1 to 0x8FFF, according to ARProperties.PullModuleAllowed), and
  - index (0 to 0x7FFF).
- IO Data as an IO combined data structure, whereas the structure is depending on general station description, each IO data structure includes status information, and addressed by
  - slot number (0 to 0x7FFF), and
  - subslot number (0 to 0x7FFF or else 1 to 0x7FFF, according to ARProperties.PullModuleAllowed)

    NOTE 1 The special subslots used for interface and related ports do not contain IO data.
- Diagnosis contains the channel or generic diagnosis for further identification, and addressed by
  - slot number (0 to 0x7FFF),
  - subslot number (0 to 0x8FFF or else 1 to 0x8FFF, according to ARProperties.PullModuleAllowed), and
  - index (implicitly set by the diagnosis service),
- Alarms, addressed by
  - slot number (0 to 0x7FFF),
  - subslot number (0 to 0x8FFF or else 1 to 0x8FFF, according to ARProperties.PullModuleAllowed), and
  - alarm type.

8.1.4.5.4 Channel

The application layer uses slots and subslots to reflect the structuring of functions and components (e.g. hardware units, logical units) inside a IO device. A slot/subslot may have

one or more channels which represent the real structure of the Input and/or Output Data. These channels may be a further subdivision of the Input and/or Output Data object.

Channels are identified in diagnosis or alarm information.

8.1.5 Instance model and device addresses

The design and construction of physical automation devices is beyond the scope of this specification and therefore vendor specific. However, Annex A provides a model for physical devices to host one or more IO controller or IO devices.

8.1.6 Application process

8.1.6.1 Overview

In the application process environment, an application may be partitioned and distributed to a number of devices on the network. Each of these partitions is referred to as an application process (AP). A device may have several APs as shown in Figure 81. In this case each individual AP is uniquely identified by an AP Identifier (API).

Each IO device shall support a Default AP with API = 0.

The Default APs shall provide device related information. Other APs are reserved to be uniquely assigned to device profiles and further standardized use.

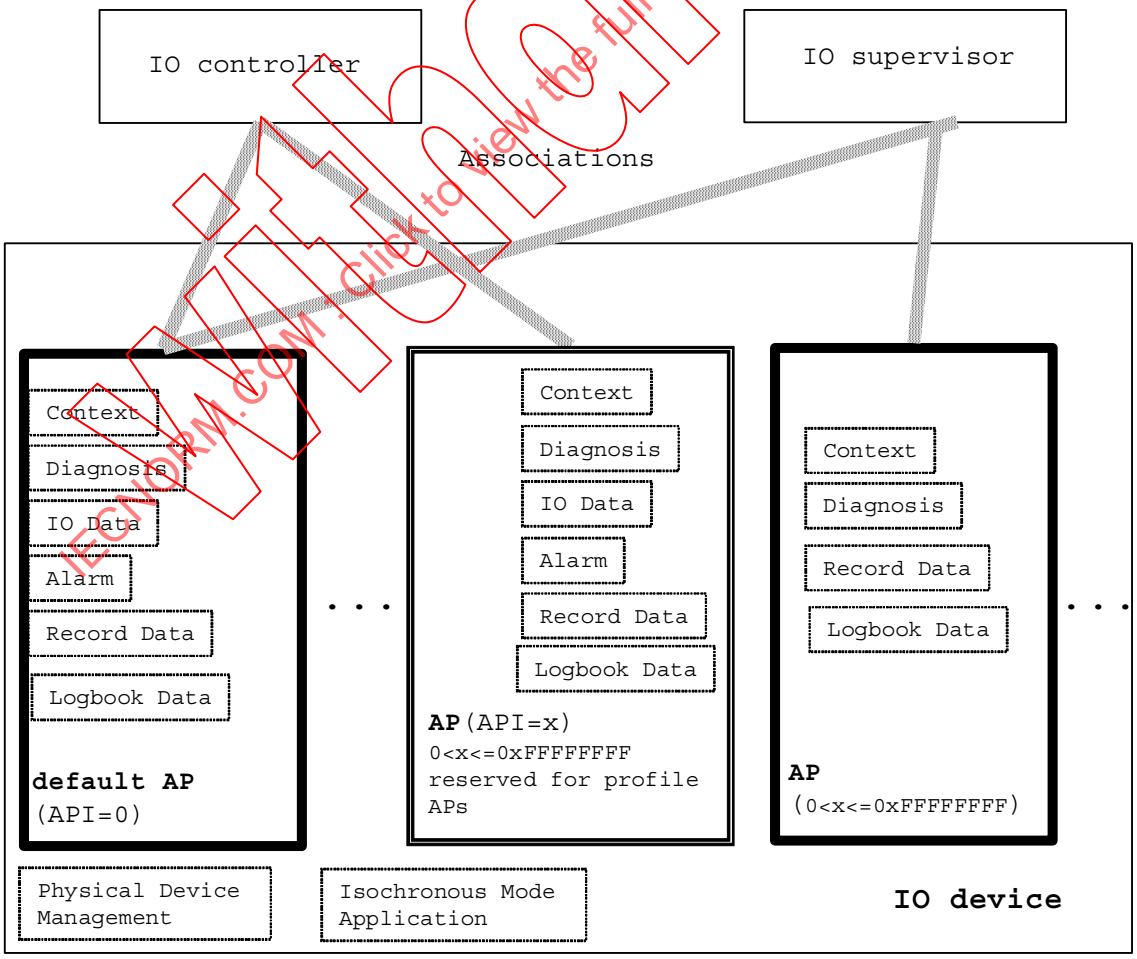


Figure 81 – Overview of application processes

An AP may be distributed to several slots and subslots. Figure 82 shows the relationship between the APs, the data elements, the slots, and subslots of an IO device. The gray boxes illustrate that there exist no "real" subslot 0. Subslot 0 shall also not contain e.g. IO data, as shown in Figure 82.

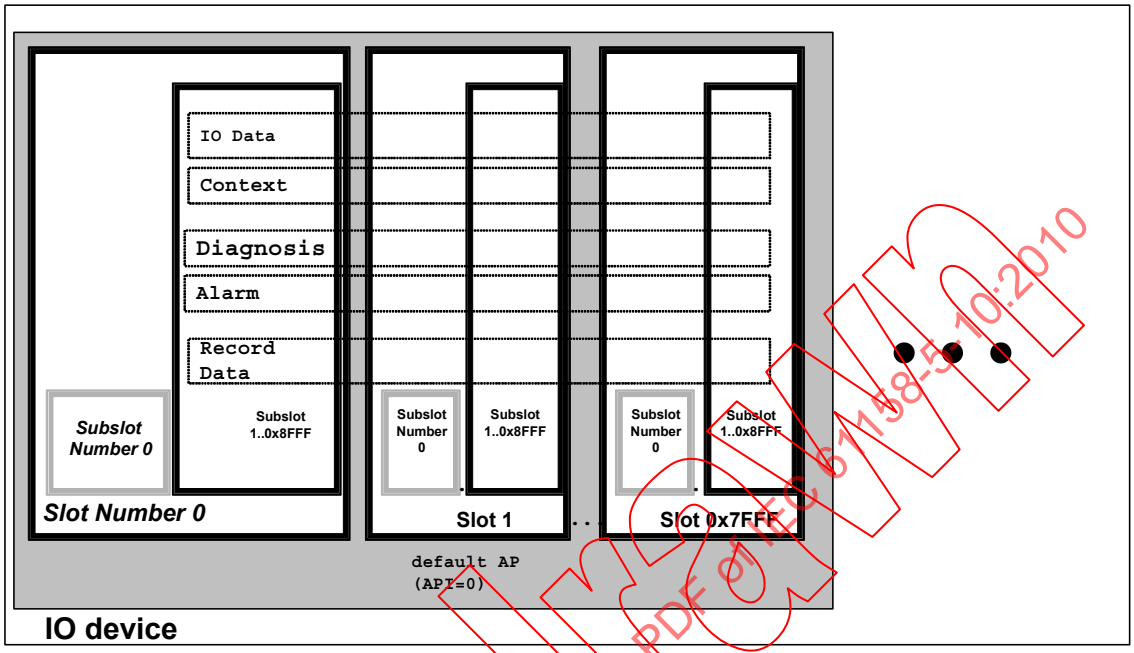


Figure 82 – IO device with APs, slots and subslots

The Subslot 0x8000 - 0x8FFF shall exist only in API 0.

NOTE 1 The mapping of a real object to the addressing model is a local matter. It means that one real object can be mapped or grouped in different APs and subslots. However, from the network perspective all address information API/slot/subslot are unique. Each API/slot/subslot can provide a different but consistent view on network visible objects. However, the IO device avoids conflicts concerning multiple use of output data referred to as overlapping outputs. An application association from a supervisor is always able to take control over output data and the controller device is informed about this by means of a special alarm.

The following rules shall be applied for the AP:

- Device instance identification data as part of the Context ASE shall always be identical regardless of the addressed AP
- API 0 addresses the application process
- API 1 to 0xFFFFFFFF is reserved for IO profiles which shall be unique.

NOTE 2 The API numbers for IO profiles are assigned by PROFIBUS International (PI).

Besides the vendor and profile specific use of the address model the following rules shall be applied for the Index:

- Index 0 to 0x7FFF shall be used by the device manufacturer for user specific Record Data Objects within the scope of the addressed AP. For API 0 indexes are vendor specific. For other APs further definitions specified in the related profile shall be applied.

NOTE 3 Index 0-0xFF can be used for migration of IEC 61784-1 CP3/1 conform devices to access the same Process Data Objects with the same index.

- Index 0x8000 to 0xFFFF are reserved for protocol internal use or further extensions and shall not be used by the application regardless of the AP.

Besides the vendor and profile specific use of the address model the following rules shall be applied for the slot:

- Slot 0 to 0x7FFF is free usable within the scope of the addressed AP. For API 0 slots are vendor specific. For other APs further definitions specified in the related profile shall be applied.
- Slot 0x8000 to 0xFFFF are reserved for future use and shall not be used regardless of the AP.

Besides the vendor and profile specific use of the address model the following rules shall be applied for the Subslot:

- Subslot 0 is addressing the module within the scope of the Pull Alarm to signal pulling of a module, if ARProperties.PullModuleAlarmAllowed is not used. If ARProperties.PullModuleAlarmAllowed is used, Subslot 1 to 0x7FFF is freely usable within the scope of the addressed AP.
- Subslot 1 to 0x7FFF is freely usable within the scope of the addressed AP. For API 0 subslots are vendor specific. For other APIs further definitions specified in the related profile shall be applied.
- For Slot 0 to 0x7FFF, Subslot 0x9000 to 0xFFFF are reserved for future use and shall not be used regardless of the AP.
- Subslot 0x8000 to 0x8FFF shall used to address the interface and port submodule. They may distributed over all slots but there shall no double use of one Subslot number across all slots. They shall only used in API 0.

Each AP shall be addressed by an APIs (x) with 0<x≤0xFFFFFFFF. The usage of additional APs shall be restricted to official profile definitions.

8.1.6.2 Application service element

An application service element (ASE), as defined in ISO/IEC 9545, is a set of application functions that provide a capability for the interworking of application processes for a specific purpose. ASEs provide a set of services for conveying requests and responses to and from application processes and their objects.

The application layer offers the following ASEs:

IO Data ASE
The IO Data ASE provides a set of services to convey IO data cyclically. These data always belong to those slots/subslots that have been configured in terms of the Context ASE. IO Data Objects contain a status for transmission. Optionally the IO Data ASE offers the possibility to share the Input Data of one IO device with other IO devices. IO Data may also be read and written by a IO supervisor acyclically.

Record Data ASE
The Record Data ASE provides a set of services to convey acyclically data. The application of the IO controller or IO supervisor requests each transmission individually. The Record Data ASE can be related to all APs of an IO device.

Logbook Data ASE
The Logbook Data ASE provides a set of services to read logbook data. The application of the IO controller or IO supervisor requests each transmission individually. The Logbook Data ASE can be related to all APs of an IO device.

Diagnosis ASE
The Diagnosis ASE provides services for the IO controller or IO supervisor to read Diagnosis information from an IO device.

Alarm ASE
The Alarm ASE provides a set of services to convey alarms issued by the IO device or IO controller. The assigned IO controller or IO device acknowledges the alarm.

Context ASE
The Context ASE provides a set of services to

- convey device parameter and configuration according device description
- identify AR endpoints
- maintain AR and CR parameter (timeouts, modes, ..)
- and to establish or release an association between individual APs.

Time ASE
The Time ASE provides a service to synchronize the network time of several or all devices on a network.

Isochronous Mode Application ASE
The Isochronous Mode Application ASE provides a set of services to parameterize and synchronize isochronous application processes.

Application Relationship ASE
The Application Relationship ASE (AR ASE) provides a description model for the separate AR types. This includes their transfer characteristics as well as their current communication states.

Physical Device Management ASE
The Physical Device Management ASE provides a set of services to manage the start-up of the physical device. It includes the assignment of a *Station Name* and further IP address parameter.

NOTE The term physical device implies that there is only one instance of this ASE for a "real" automation device regardless of the number of instances of device types within the automation device. Anyway, it will be only one instance for most devices.

8.1.6.3 Application process objects

An application process object (APO) is a network representation of a specific aspect of an application process (AP). Each APO represents a set of information and processing capabilities of an AP that are accessible through services of the application layer. APOs are used to represent these capabilities to other APs in a system.

In order to permit an AP to communicate with an AP of another device, APOs have to be available. Application process objects (virtual objects) represent existing process objects (real objects), that an application process has made visible and accessible to the communication (see Figure 83).

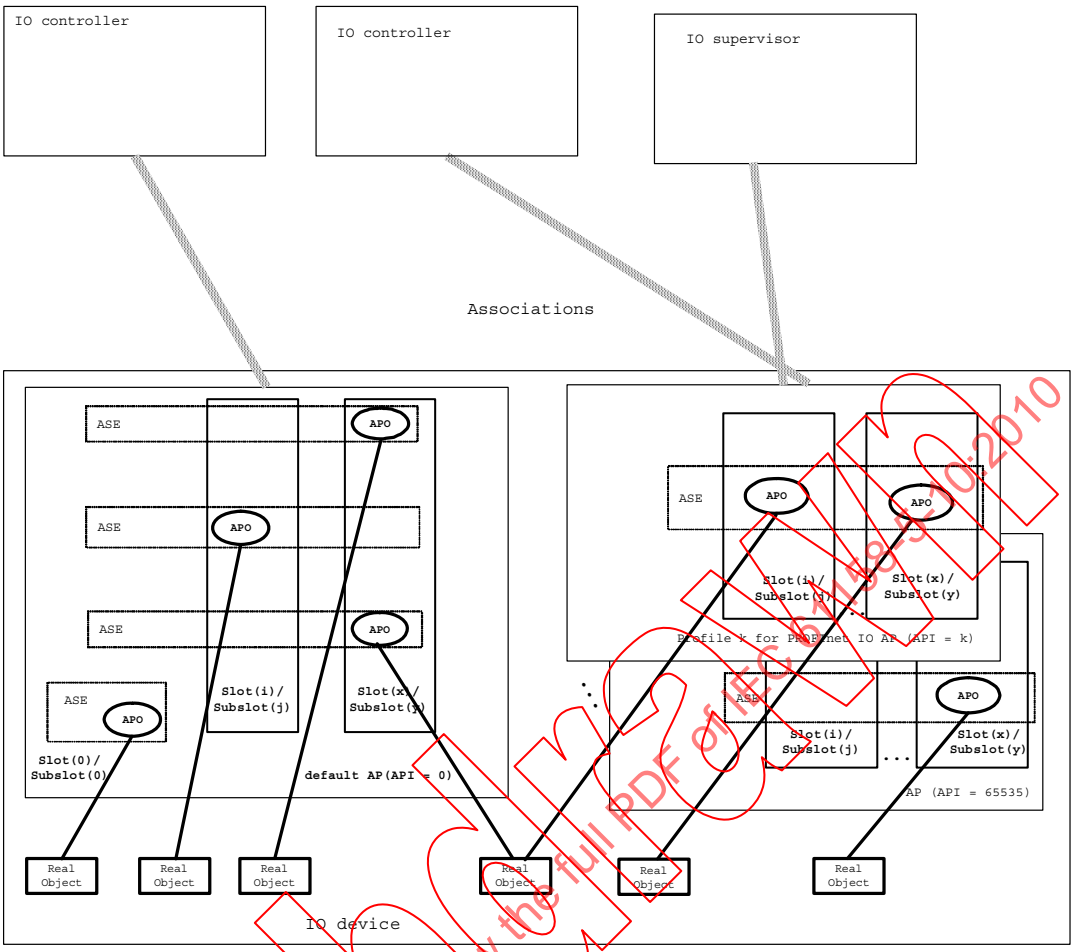


Figure 83 – Application Process with application process objects (APOs)

In Figure 84, a remote AP acting as Client may access the real object by sending requests through the APO that represents the real object. Local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

Within the AP an APO is identified by slot, subslot and index. The address space which is defined by slot, subslot and index can be used by several APs.

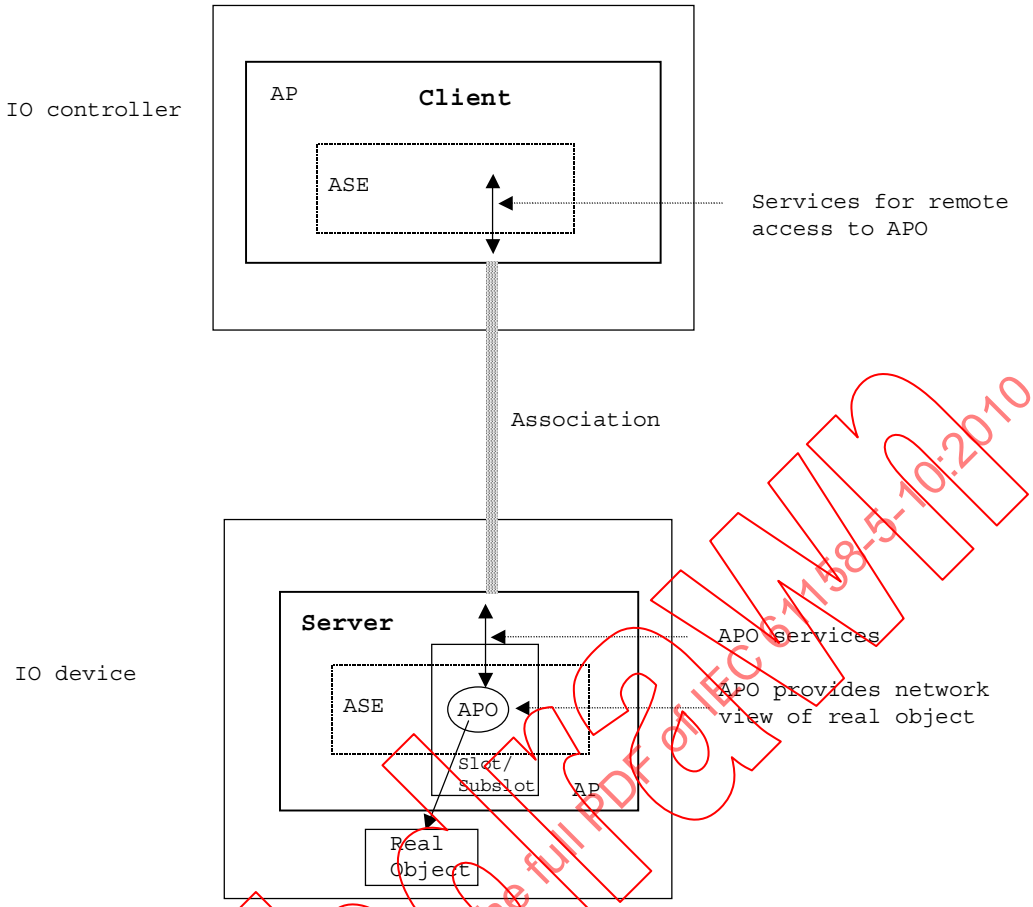


Figure 84 – Access to a remote APO

In Figure 85, a Client/Server and a Provider/Consumer association is shown. The Client may access the real object by sending requests through the APO that represents the real object. The Consumer may subscribe to the whole or to a part of the remote APO. The Provider/Consumer association is always related to the IO Data ASE. Local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

NOTE The mapping of real data to network visible APOs is a local matter. However, the device vendor has to provide local means to avoid conflicts with overlapping outputs and writeable variables.

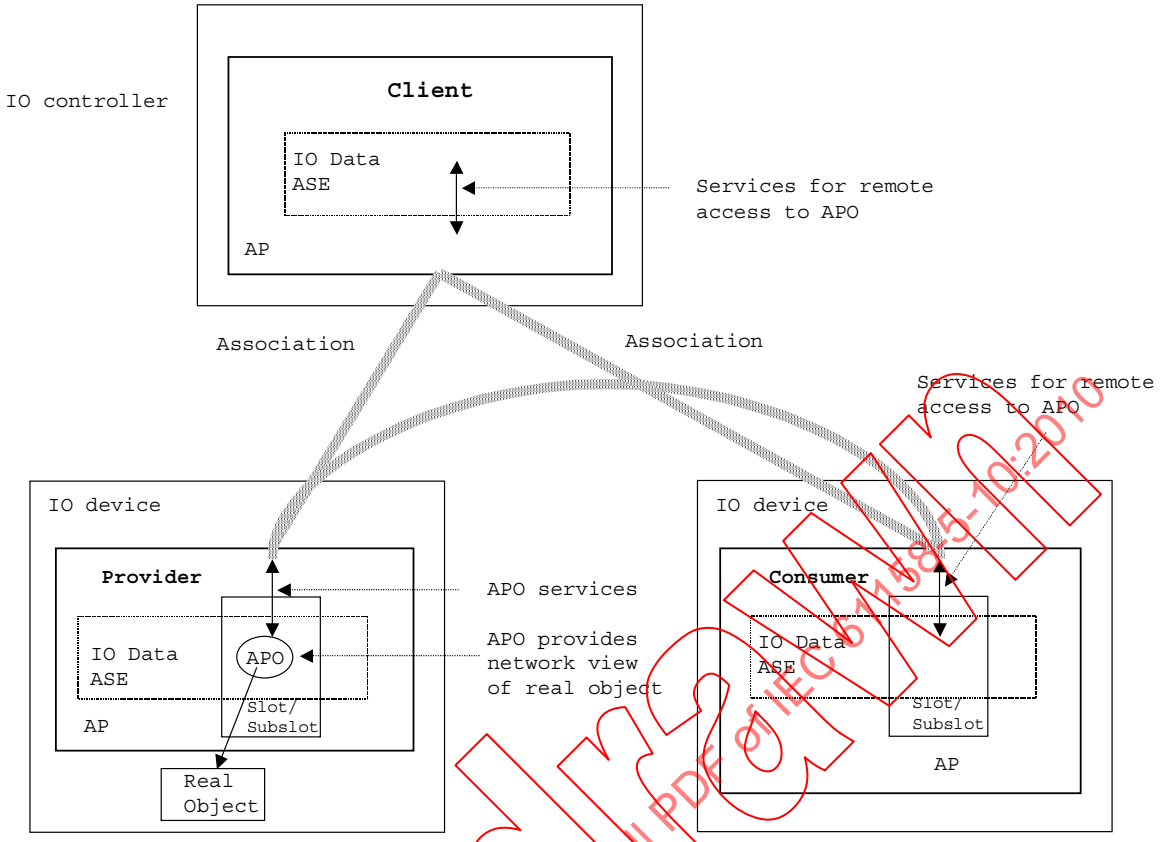


Figure 85 – Access to a remote APO for provider/consumer association

8.1.7 Application relationship

8.1.7.1 General

An Application Relationship (AR) is a co-operative relationship between two or more APs for the purpose of exchange of information and co-ordination of their joint operation (see Figure 86). This relationship is activated by the exchange of Application Protocol Data Units (APDU). The application layer uses different types of ARs which distinguish in their conveyance characteristic.

8.1.7.2 Application relationship endpoint

An AP has access to the communication using application relationship endpoints (AREP, see ISO 7498-1). One or more AREPs are fixed and uniquely assigned to an AP. These endpoints are addressed by the AP through an identifier (AREP ID). These identifiers are device specific and not defined by the communication itself. ARs are defined as a set of co-operating AREPs. Between two APs one or more ARs may exist, each one having unique AREPs. Figure 86 shows an example of one AR with two AREPs.

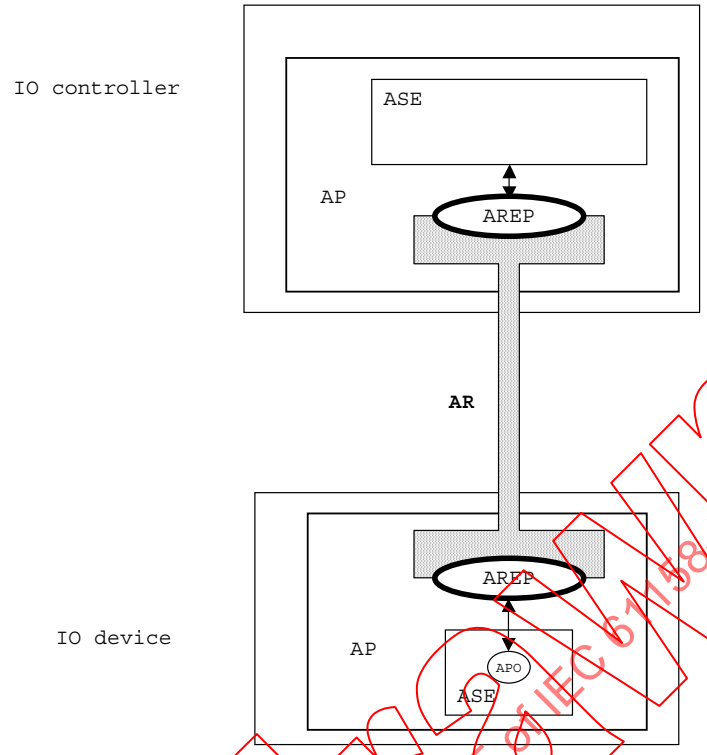


Figure 86 – Example of one AR with two AREPs

8.1.7.3 Overview of application relationships

The application layer offers the following AR types:

NOTE The AR can be identified with a unique UUID referred to as AR UUID.

IO AR: application relationship between the application process of one IO controller and the related application process of an IO device for the following purposes:

- cyclic exchange of the Input data with the IO controller by means of Input CR (BBUU CR)
- cyclic exchange of the Output data with the IO controller by means of Output CR (BBUU CR)
- cyclic exchange (multicast) of the Input data or Output data by means of M CR (BBUU-OM CR)
- acyclic data transfer for e.g. parameterization, configuration, IO data and diagnosis (Record Data objects) by means of RDCM CR (QQCB-CO CR)
- transmission of alarm data in both directions by means of Alarm CR (QQCB-CO CR)

The IO AR shall only be used together with API 0 to 0xFFFFFFFF.

Supervisor AR: application relationship between:

- the application processes of one IO supervisor and one IO device,
- the application processes of one IO parameter server and one related IO device,
- the application processes of one IO controller and one IO device,

for the following purposes:

- same functions as provided by IO AR with API 0 to 0xFFFFFFFF,
- additional take over of an IO AR (controlled by supervisor)

Implicit AR: (AR UUID = UUID_NIL) application relationship between:

- the application processes of IO controller, IO devices, IO supervisor to Read ASE object values

The Implicit AR can be used together with all APIs.

8.2 ASE data types

The data types supported by decentralized periphery are a subset of those defined in clause 5. They are:

Boolean
Date
TimeOfDay
TimeOfDay with date indication
TimeOfDay without date indication
TimeDifference
TimeDifference with date indication
Float32
Float64
Integer8
Integer16
Integer32
Integer64
Unsigned8
Unsigned16
Unsigned32
Unsigned64
UUID
NetworkTime
NetworkTimeDifference
OctetString
VisibleString

8.3 ASEs

8.3.1 Record data ASE

8.3.1.1 Overview

In the Application Layer environment, application processes contain data that remote applications are able to read and write. The Record Data ASE defines attributes of general purpose Record Data objects and provides a set of services used to read, and write their values. Furthermore, the client may cancel a requested outstanding service due to special application conditions such as dynamic reconfiguration. A Record Data ASE is specific for an AP and each AP shall contain one Record Data ASE. Record Data objects within an AP are addressed by Slot Numbers, Subslot Numbers, and Index. The Record Data object can be

read and written partially or entirely. The local connection between a Record Data object and a real world or physical object is not in the scope of this definition.

NOTE A Record Data object may be connected to one real world object for read and write, or, it can be assigned to one real object for reading and for another one for writing the value.

The IO controller and IO supervisor are able to read or write the value of the Record Data object within the AP of an IO device. Furthermore, an IO device is able to read or write the value of a Record Data object within the AP of an IO parameter server. The IO supervisor is able to read or write the value of a Record Data object within the AP of an IO controller. Using the attribute Device Access an IO supervisor is able to read or write all Record Data objects if permitted by the application process of the IO device.

The client application process has to use the Context ASE to establish an association to get access to the Record Data objects of a server. Read access is also possible without a connection via the Implicit AR.

The access to Record Data objects is performed according to the Client/Server access model. The Client/Server model is characterized by a Client application sending a read or write request to a Server application that responds accordingly.

The formal model of the Record Data ASE is presented by the Record Data class specification, containing a description of its attributes, services as well as invocations, followed by a detailed service specification.

Special Record Data objects shall be reserved for identification and maintenance functions. The reserved indices are 0xAFF0 to 0xAFFF. The object on index 0xAFF0 shall be read only.

Furthermore, additional reservations are defined for further Profile definitions.

8.3.1.2 Record data class specification

8.3.1.2.1 Template

A Record Data object may be related to real object as shown in Figure 87.

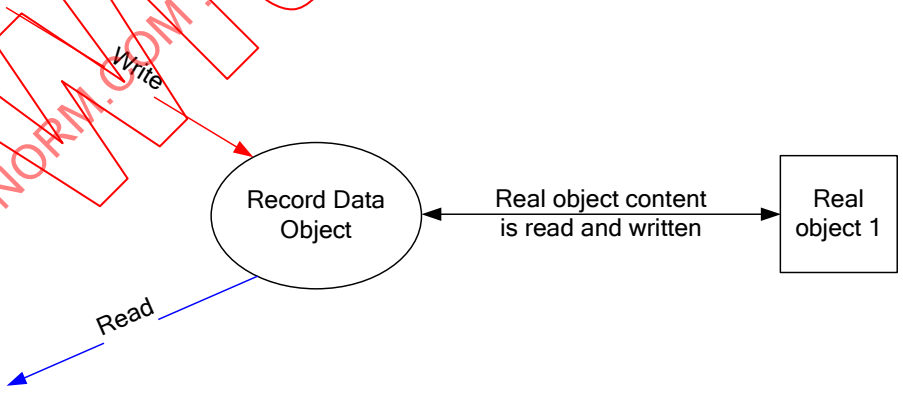


Figure 87 – Relation of a record data object to one real object

A Record Data object may be related to two or more real objects as shown in Figure 88.

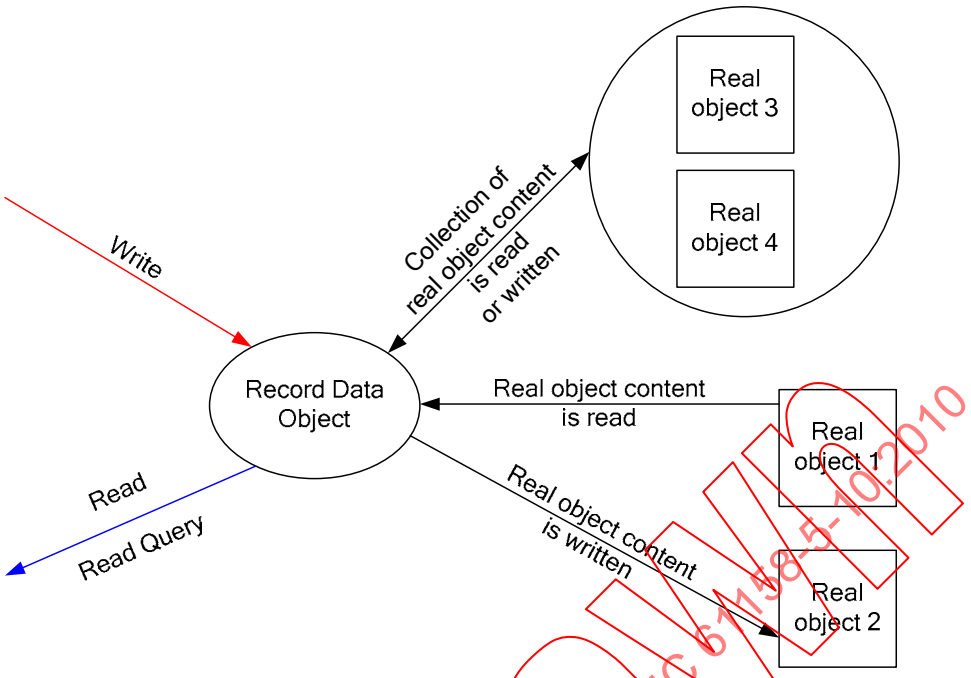


Figure 88 – Relation of a record data object to two real objects

A Record Data object is described by the following template:

ASE:	**Record Data ASE**	
CLASS:	**Record Data**	
CLASS ID:	**not used**	
PARENT CLASS:	**TOP**	
ATTRIBUTES:		
1	(m) Key Attribute:	Identifier
2	(m) Attribute:	Read Record Data Description
3	(m) Attribute:	Read Partial Access
4	(m) Attribute:	Write Record Data Description
5	(m) Attribute:	Write Partial Access
6	(m) Attribute:	Shared Write Access
7	(o) Attribute:	Write Persistence Flag
SERVICES:		
1	(m) OpsService:	Read
2	(o) OpsService:	Read Query
3	(m) OpsService:	Write

8.3.1.2.2 Attributes

Identifier

This key attribute is composed of API, Slot Number, Subslot Number, and Index to define to which Application Process Identifier, structural elements slot (module), subslot (submodule), and to which index the Record Data object belongs. This Identifier is unique in the IO device and shall not be used by another object.

Attribute type: Unsigned32, Unsigned16, Unsigned16, Unsigned16

Allowed values for API: 0, all others are reserved for Profile definitions

Allowed values for Slot Number and Subslot Number: 0 to 0x7FFF, additionally Subslot Number 0x8000 to 0x8FFF are allowed

NOTE 1 Subslot Number 0 is used in conjunction with AlarmType “Pull” to address a certain module within the IO device.

Allowed values for Index: 0 to 0x7FFF, 0xAFF0 to 0xAFFF for I&M Record Data objects (0xAFF0 read only),
reserved for profile definitions are: 0xB000-0xBFFF, 0xD000-0xDFFF, 0xEC00-0xEFFF, 0xF400-0xF7FF, 0xFC00-0xFFFF.

Other values are reserved for further use.

Read Record Data Description
This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in IEC/TR 61158–1:2010 is used.

Read Partial Access
This attribute defines whether a partial read access to the Record Data object is supported or not. This attribute controls the behavior of the Read service. Partial access equal TRUE means that the content of the Record Data object may be read from the first octet up to the octet defined by the length given in the service request even if the length is shorter than the length of the Record Data object. The entire object will be read if the length given in the service is greater or equal than the length of the object in any case. Partial access equal FALSE means that the content of the Record Data object can only be read entirely. That implies that the length given in the service shall be equal or greater to the length of the object. Otherwise it shall return a negative response with the error code "Invalid type".

Attribute type: Boolean

Write Record Data Description
This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in IEC/TR 61158–1:2010 is used.

Write Partial Access
This attribute defines whether a partial write access to the Record Data object is supported or not. This attribute controls the behavior of the Write service. Partial access equal TRUE means that the content of the Record Data object shall be written from the first octet up to the octet defined by the length given in the service request even if the length is shorter than the length of the Record Data object. The entire object will be written if the length given in the service is equal to the length of the object in any case. Partial access equal FALSE means that the content of the Record Data object can only be written entirely. That implies that the length given in the service shall be equal to the length of the object. Otherwise it shall return a negative response with the error code "Write Length Error". If the length in the Write service exceeds the length of the Record Data object a negative response with the error code "Write Length Error" shall be returned.

Attribute type: Boolean

Shared Write Access
This attribute defines whether a write access to the Record Data object is supported for client connected via a shared application relationship. This attribute controls the behavior of the Write service. Shared Write Access equal TRUE means that the content of the Record Data object can be written from client connected via a shared application relationship. In this case, the local application is responsible for data consistency. Shared Write Access equal FALSE means that the content of the Record Data object shall not be written from client connected via a shared application relationship.

Attribute type: Boolean

Write Persistence Flag
This optional attribute shall define the persistence behavior for the value of the Record Data Object during a power cycle. It shall be stored non-volatile if the value is set to TRUE. Otherwise the written value is volatile and after a power cycle missing.

NOTE 2 This attribute is a local device property implemented on behalf of the device vendor.

The application process shall implement the persitence storage behavior of Table 243 according to this attribute and the Write indication service parameter Prm Flag.

Table 243 – Persistence behavior for record data objects

Write Persistence Flag	Write.ind (Prm Flag)	Persistence Storage
FALSE	FALSE	shall be stored volatile
FALSE	TRUE	shall be stored volatile
TRUE	FALSE	shall be stored non-volatile before Write.rsp(+)
TRUE	TRUE	Write.ind: Shall be only intermediately stored volatile and shall immediately respond Write.rsp(+) before receiving Prm End indication Application Ready.req: Before issueing the Application Ready request, all intermediate stored data shall be checked for consistency, only consistent data set shall be stored non-volatile, shall checked again for consistency in non-volatile memory, and send Application Ready request afterwards
NOTE Only consistent sets of data within the parameterization phase are stored non-volatile. An AR failure during this phase does not affect previously stored record data and the intermediate buffer is deleted.		

NOTE 3 An application process may avoid persistence saving of unchanged values by local means, e.g. using checksums.

Attribute type: Boolean

8.3.1.2.3 Invocation of the Record Data object

For the invocation of the Record Data object the following rules apply:

- Record Data objects shall not exceed the total length of 2^{32}-65 octets. The attribute Record Data description has to be set accordingly.
- At least one service shall be allowed for the access of the object.
- At least one AR shall have access to the object.
- The access rights have to be set accordingly to the allowed service(s).

8.3.1.3 Record data service specification

8.3.1.3.1 Read

This confirmed service may be used to read the value of a Record Data object. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 244 shows the parameters of the service.

The attribute Read Partial Access controls the behavior of the response in case the requested length is shorter than the length of the Record Data object (see attribute description).

Table 244 – Read

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Index	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)
Length			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired Record Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired Record Data object in a specific subslot (typically a submodule).

Index
The parameter Index is used in the destination device for addressing the desired Record Data object.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of a Record Data object that has to be read. The allowed length is from 0 to 2^{32}-65.

Result(+)
This parameter indicates that the service request succeeded.

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 can be used by profile specifications to transmit specific messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 can be used by device vendors to transmit specific additional messages.

Data
The parameter Data contains the value of the object which has been read and consists of the number of octets indicated in the Length of the response. This parameter has to be composed of the data types defined in Clause 5.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.1.3.2 Read Query

This optional confirmed service may be used to read the value of a Record Data object. To optimize the access to many small data objects a server may offer different combinations of this data behind one Index. In this case, the content of the Record Data object is defined by

the Selector. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 245 shows the parameters of the service.

The attribute Read Partial Access controls the behavior of the response in case the requested length is shorter than the length of the Record Data object (see attribute description).

Table 245 – Read Query

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Index	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Selector	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)
Length			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired Record Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired Record Data object in a specific subslot (typically a submodule).

Index
The parameter Index is used in the destination device for addressing the desired Record Data object.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding

service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of a Record Data object that has to be read. The allowed length is from 0 to 2^{32}-65.

Selector
The parameter Selector is used by the server to identify the requested data. To optimize the access to many small data objects a server may offer different combinations of this data behind one Index and select the requested collection with the Selector.

Result(+)
This parameter indicates that the service request succeeded.

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 can be used by profile specifications to transmit specific messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 can be used by device vendors to transmit specific additional messages.

Data
The parameter Data contains the value of the object which has been read and consists of the number of octets indicated in the Length of the response. This parameter has to be composed of the data types defined in Clause 5.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.1.3.3 Write

This confirmed service may be used to write the value of a Record Data object. This service shall only be used in conjunction with the IO AR or Supervisor AR. Table 246 shows the parameters of the service.

The attribute Write Partial Access controls the behavior of the response in case the requested length is shorter than the length of the Record Data object (see attribute description).

The service parameter “Multiple” may be used to convey multiple Record Data objects within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE 1 It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE) in one APDU.

Table 246 – Write

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Index	M	M(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Data	M	M(=)		
Prm Flag		M		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired Record Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired Record Data object in a specific sub-element (typically a submodule).

Index
The parameter Index is used in the destination device for addressing the desired Record Data object.

Multiple
The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of a Record Data object which has to be written. The allowed length is from 0 to 2^{32}-65.

Data
The parameter Data contains the value of the Record Data object which has to be written and consists of the number of octets indicated in the Length of the request. This parameter has to be composed of the data types defined in Clause 5.

Prm Flag
The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

NOTE 2 This parameter is used to control the persistent storage of data. It is utilized in fast start-up procedure.

Result(+)
This parameter indicates that the service request succeeded.

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 can be used by profile specifications to transmit specific response messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 can be used by device vendors to transmit specific response messages.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: write error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, write constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 5 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 6 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.2 IO Data ASE

8.3.2.1 Overview

In the application layer environment, each application process of an IO device shall contain exactly one IO Data object for each application process instance. It is structured by means of slots and subslots. Subslots represent IO data that shall be cyclically conveyed over the network. Therefore the IO Data ASE defines attributes of the IO Data objects and provides a set of services used to get and set the buffer to transport their values. The IO Data objects are composed of an Input Data object and an Output Data object.

An Input Data object or an Output Data object shall be partitioned in Input Data or Output Data Elements which represent the IO Data of the slot/subslot.

Additional services are provided to read acyclically the values of the IO Data objects and to indicate new values for the Input Data object and the Output Data object. Furthermore services are provided to indicate the presence and to get the content of new consumed Input Data.

The IO Data objects are implicitly addressed through the related services. The granularity of input or output data in a Server/Provider is according to the correspondent configuration attributes.

The I/O Data ASE uses the Client/Server and Provider/Consumer access model. The Client/Server model is characterized by a Client application conveying the value for the Output Data object to the Server's buffer. The Server application fetches this value by a Get Output service. The receipt of a new value is indicated by the New Output service. The Server application conveys the value of it's Input Data object via the Set Input service to the Server's buffer as soon as one of the real object values changes. The Client application gets this value by the Get Input service. The transmission of the Server buffer to the Client buffer and vice versa is done asynchronously to the Get Output and Set Input services.

The formal model of the IO Data ASE is described next, followed by a description of its services. Furthermore, the IO Data ASE represents the real input and output structure of an IO device. The relations between real IO data structure and conveyance of IO data via ARs are described within the Context ASE.

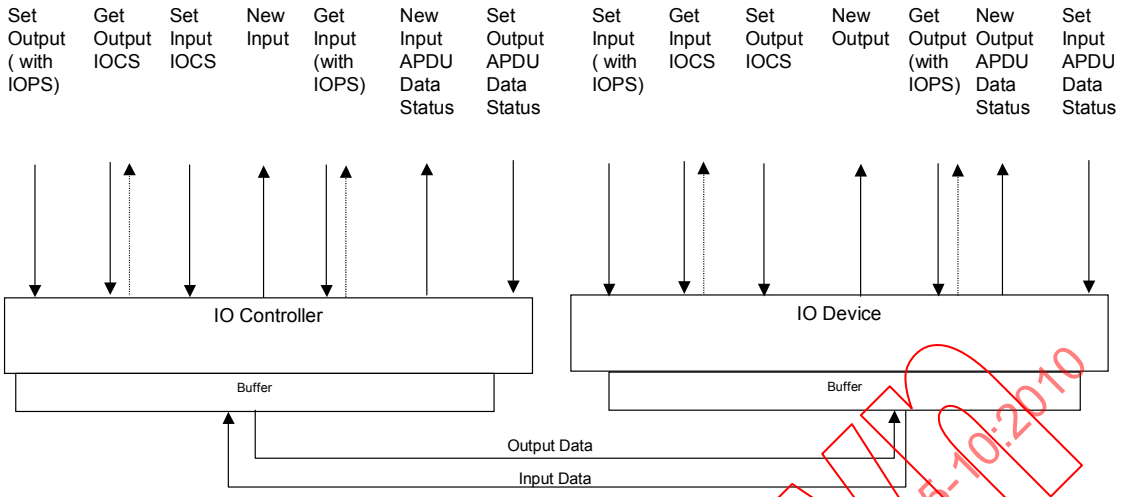


Figure 89 – Overview IO ASE service interactions

Figure 89 shows the IO service interactions for the IO controller and IO device. The services interact with local buffers maintained by the application layer protocol. This protocol provides mechanisms for cyclic exchange of buffers between the devices. Local services provide a means to get or set the values of IO data together with their status values. The status values are referred to as IOPS and IOCS. The IOPS (Input Output Provider Status) characterizes the status of the data source. It belongs to Output Data for an IO controller and to Input Data for an IO device. The IOCS (Input Output Consumer Status) characterizes the feedback of the data sink. It belongs to Input Data for an IO controller and to Output Data for an IO device. Furthermore, services are provided to set and indicate a change of the application data status (APDU Data Status).

NOTE If a submodule does not contain any input or output data it is considered as an Input Data item with the length of Input Data attribute set to zero.

The transition of IOPS/IOCS from GOOD to BAD may be done without further notification (e.g. alarms). The transition of IOPS/IOCS from BAD to GOOD shall be notified to and acknowledged from the IO Controller by means of an alarm after a change of IOXS from Bad->Good in data exchange state.

8.3.2.2 IO Data class specification

8.3.2.2.1 Input Data class specification

8.3.2.2.1.1 Template

The Input Data object is described by the following template:

ASE:	**IO Data ASE**		
CLASS:	**Input Data**		
CLASS ID:	**not used**		
PARENT CLASS:	**TOP**		
ATTRIBUTES:			
1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	List of APs
2.1	(m)	Attribute:	API
2.2	(m)	Attribute:	List of Input Data Elements
2.2.1	(m)	Attribute:	Slot Number
2.2.2	(m)	Attribute:	List of Subslots
2.2.2.1	(m)	Attribute:	Subslot Number
2.2.2.2	(m)	Attribute:	Input Data Object Format

2.2.2.2.1	(m)	Attribute:	Length Input Data
2.2.2.2.2	(m)	Attribute:	Input Data Structure
2.2.2.3	(m)	Attribute:	Length IOPS
2.2.2.4	(m)	Attribute:	IOPS
2.2.2.5	(m)	Attribute:	Length IOCS
2.2.2.6	(m)	Attribute:	IOCS

SERVICES:

1	(m)	OpsService:	Set Input
2	(m)	OpsService:	Set Input IOCS
3	(m)	OpsService:	Get Input
4	(m)	OpsService:	Get Input IOCS
5	(o)	OpsService:	New Input
6	(m)	OpsService:	Set Input APDU Data Status
7	(m)	OpsService:	New Input APDU Data Status
8	(m)	OpsService:	Read Input Data

8.3.2.2.1.2 Attributes

Implicit

The attribute Implicit indicates that the Input Data object is implicitly addressed by the services.

List of APs

One AP is composed of the following list elements:

API

This attribute defines to which application process the Input Data Elements belongs.

Attribute type: Unsigned32

Allowed values: 0 - default application process. The values 1 to 0xFFFFFFFF are reserved for future use in profile guidelines.

List of Input Data Elements

One Input Data Element is composed of the following list elements:

Slot Number

This attribute defines to which module the Input Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Subslots

This attribute consists of the following attributes.

Subslot Number

This attribute defines to which subslot the Input Data Element belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE Subslot Number 0 is used in conjunction with AlarmType “Pull” to address a certain module within the IO device. It is never a “real” submodule and it does not contain Input Data Elements.

Input Data Object Format

This attribute consists of the following attributes.

Length Input Data

The attribute Length Input Data defines the number of octets of the Input Data Element without IOPS.

Attribute type: Unsigned16

Allowed values: 0 to 1 439

Input Data Structure
This attribute contains either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in IEC/TR 61158–1:2010 is used.

The Input Data shall be transmitted consistent from the sending buffer to the receiving buffer completely. However, the application may get or set only parts of a single data object including the IOPS.

This field is part of the GSDML.

Furthermore, if the Input Data are the source of a Multicast Provider CR then the associated Output Data of the Multicast Consumer CR shall have exactly the same data structure.

Length IOPS
The attribute Length IOPS defines the number of octets of the provider status.

Attribute type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOPS
The attribute defines the provider status for the Input Data Object Element. It defines the status of the Input Data Object Element which belongs to the subslot of the IO device. It provides a means to identify the first instance which detected the invalid content of the Input Data Object Element by local means. The detecting instance shall set the IOPS accordingly and immediately. Possible instances are the submodule, the module, the IO device, and the IO controller (locally, e.g. watchdog timeout). Only valid data with IOPS==GOOD shall be processed by the application. In all other cases the application shall use default values (e.g. zero, last valid value, substitute value) according to its configuration and parameterization.

Attribute type: Unsigned8

Allowed values:
GOOD – content of Input Data Object Element is valid,
BAD_BY_SUBSLOT - content of Input Data Object Element is invalid, it was detected by the submodule,
BAD_BY_SLOT - content of Input Data Object Element is invalid, it was detected by the module,
BAD_BY_DEVICE - content of Input Data Object Element is invalid, it was detected by the IO device,

Special case: BAD_BY_CONTROLLER - content of Input Data Object Element is invalid, only locally detected by the IO controller itself e.g. through a communication timeout. This value shall only be used with the service parameter IOPS together with the service Get Input at the IO controller. It shall never be applied with the Set Input service at the IO device.

Length IOCS
The attribute Length IOCS defines the number of octets of the consumer status.

Attribute type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOCS
The attribute defines the consumer status for the Input Data object element as an application feedback of the IO controller. It is not only a mean to detected communication problems. Moreover it is a mean to inform the IO device that the control application or parts of the control application are not processing the values. Therefore the application process shall set the IOCS to BAD_BY_CONTROLLER if the Input Data Object Element could not be processed e.g. if the application program stopped. The possible action of the IO device is beyond the scope of this service definition. The attribute IOCS shall only contain a valid value if the Input

Data Object Element is part of an established Application Relationship (AR). In all other cases the value of this attribute has no meaning.

Attribute type: Unsigned8

Allowed values:
GOOD - the Input Data Object Element could be successfully processed by the application process of the IO controller,
BAD_BY_CONTROLLER - the Input Data Object Element could not be successfully processed by the application of the IO controller e.g. in case of an error or as cause of the operation state („stop“) of the IO controller

Special case: BAD_BY_DEVICE – the IO device has locally detected problems to convey data. It shall never be applied with the Set Input IOCS service at the IO controller.

8.3.2.2.1.3 Invocation of the Input Data object

For the invocation of the Input Data object the following rules apply:

- Input Data object elements data shall not exceed the total length of 1 439 octets. The attributes List of Input Data Elements, Input Data Object Format respectively Format and Length have to be set accordingly.
- The relation of an Input Data object element to a Input CR, Output CR (only IOCS), M CR will be handled in the Context ASE.
- The sum of all Input Data object elements related to one Input CR or M CR including IOPS and IOCS of the related Output Data object elements shall not exceed the total length of 1 440 octets.
- One Input Data object shall be invoked in an application process of an IO device.

8.3.2.2 Output data class specification

8.3.2.2.1 Template

The Output Data object is described by the following template:

ASE: **IO Data ASE**
CLASS: **Output Data**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	List of APs
2.1	(m)	Attribute:	API
2.2	(m)	Attribute:	List of Output Data Elements
2.2.1	(m)	Attribute:	Slot Number
2.2.2	(m)	Attribute:	List of Subslots
2.2.2.1	(m)	Attribute:	Subslot Number
2.2.2.2	(m)	Attribute:	Output Data Object Format
2.2.2.2.1	(m)	Attribute:	Length Output Data
2.2.2.2.2	(m)	Attribute:	Output Data Structure
2.2.2.3	(m)	Attribute:	Length IOPS
2.2.2.4	(m)	Attribute:	IOPS
2.2.2.5	(m)	Attribute:	Length IOCS
2.2.2.6	(m)	Attribute:	IOCS
2.2.2.7	(m)	Attribute:	Safe State Behavior Changeable
2.2.2.8	(m)	Attribute:	Substitute Mode
2.2.2.9	(m)	Attribute:	Output Substitute Data Valid
2.2.2.10	(m)	Attribute:	Output Substitute Data

SERVICES:

1	(m)	OpsService:	Set Output
2	(m)	OpsService:	Set Output IOCS
3	(m)	OpsService:	Get Output
4	(m)	OpsService:	Get Output IOCS
5	(o)	OpsService:	New Output
6	(m)	OpsService:	Set Output APDU Data Status
7	(m)	OpsService:	New Output APDU Data Status
8	(m)	OpsService:	Read Output Data
9	(m)	OpsService:	Read Output Substitute Data
10	(m)	OpsService:	Write Output Substitute Data

8.3.2.2.2.2 Attributes

Implicit
The attribute Implicit indicates that the Output Data object is implicitly addressed by the services.

List of APs
One AP is composed of the following list elements:

API
This attribute defines to which application process the Output Data Elements belongs.

Attribute type: Unsigned32

Allowed values: 0 - default AP; the values 1 to 0xFFFFFFFF are reserved for the use in profile guidelines.

List of Output Data Elements
One Output Data Element is composed of the following list elements:

Slot Number
This attribute defines to which module the Output Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This attribute defines to which subslot the Output Data Element belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE Subslot Number 0 is used in conjunction with AlarmType “Pull” to address a certain module within the IO device. It is never a “real” submodule and it does not contain Output Data Elements.

Output Data Object Format
This attribute consists of the following attributes.

Length Output Data
The attribute Length Output Data defines the number of octets of the Output Data Element without IOPS.

Attribute type: Unsigned16

Allowed values: 1 to 1 439

Output Data Structure
This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in IEC/TR 61158–1:2010 is used.

The Output Data shall be transmitted consistent from the sending buffer to the receiving buffer completely. However, the application may get or set only parts of a single data object including the IOPS.

This field is part of the GSDML.

Furthermore, if the Output Data are the sink of a Multicast Consumer CR then the associated Input Data of the Multicast Provider CR shall have exactly the same data structure.

Length IOPS
The attribute Length IOPS defines the number of octets of the provider status.

Attribute type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOPS
The attribute defines the provider status for the Output Data Object Element. It defines the status of the Output Data Object Element which belongs to the submodule of the IO device. Moreover it is a means to inform the IO device that the control application or parts of the control application are not processing. The application process shall set the IOPS to BAD_BY_CONTROLLER if the Output Data Object Element could not set e.g. if the application program stopped. The IO device shall use the default values (zero, last valid value, substitute value) for process output in such a case. The attribute IOPS shall only contain a valid value if the Output Data Object Element is part of an established Application Relationship (AR). In all other cases the value of this attribute has no meaning.

Attribute type: Unsigned8

Allowed values:
GOOD - the Output Data Object Element was successfully processed by the application process of the IO controller,
BAD_BY_CONTROLLER - the Output Data Object Element could not be successfully processed by the application of the IO controller e.g. in case of an error or as cause of the operation state („stop“) of the IO controller

Special case: BAD_BY_DEVICE – the IO device has locally detected problems to convey data. It shall never be applied with the Set Input service at the IO controller.

Length IOCS
The attribute Length IOCS defines the number of octets of the consumer status.

Attribute type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOCS
The attribute defines the consumer status for the Output Data object element as an application feedback of the IO device. It is not only a means to detected communication problems. Moreover it is a means to inform the IO controller that the device application could not set the values to the “real” process for local reasons. The detecting instance shall set the IOCS accordingly and immediately. Possible instances are the submodule, the module, the IO device, and the IO controller (locally, e.g. watchdog timeout). The possible action of the IO controller is beyond the scope of this service definition. The attribute IOCS shall only contain a valid value if the Output Data Object Element is part of an established Application Relationship (AR). In all other cases the value of this attribute has no meaning.

Attribute type: Unsigned8

Allowed values:
GOOD – the Output Data Object Element was successfully processed,
BAD_BY_SUBSLOT - the Output Data Object Element was not successfully processed, it was detected by the submodule,
BAD_BY_SLOT - the Output Data Object Element was not successfully processed, it was detected by the module,
BAD_BY_DEVICE - the Output Data Object Element was not successfully processed, it was detected by the IO device,

Special case: BAD_BY_CONTROLLER - only locally detected by the IO controller itself e.g. through a communication timeout. This value shall only be used with the service parameter IOCS together with the service Get Output IOCS at the IO

controller. It shall never be applied with the Set Output IOCS service at the IO device.

Safe State Behavior Changeable
The Safe State Behavior Changeable attribute defines if this behavior is adjustable.

Attribute type: Boolean

Substitute Mode
The Substitute Mode attribute defines the handling of the output data subslot in case of a failure. This behavior shall be applied if the IOPS value contains other values than GOOD.

Attribute type: Unsigned16

Allowed Values:
ZERO – is a submodule specific function to set all output values of the Output Data element to zero (IOPS==GOOD) or inactive (IOPS==BAD),
LAST_VALUE – all output values of the Output Data element shall be set to the last valid (IOPS==GOOD) values of the Get Output confirmation service primitive,
REPLACEMENT_VALUE – all output values of the Output Data element shall be set to the value of the attribute Safe State Value

PROFILE_SPECIFIC1 - PROFILE_SPECIFIC255 reserved for profile specification

Output Substitute Data Valid
The Output Substitute Data Valid attribute defines the validity of the attribute Output Substitute Data.

Attribute type: Boolean

Allowed Values:
TRUE shall be used in conjunction with Substitution Mode ZERO, LAST_VALUE, REPLACEMENT_VALUE and PROFILE_SPECIFIC to indicate valid data.
FALSE shall be used in conjunction with Substitution Mode ZERO and PROFILE_SPECIFIC to indicate invalid data. In this case the data content shall be set to zero.

Output Substitute Data
The Output Substitute Data attribute defines the replacement value of the output data for the subslot in case of a failure.

The attribute type shall be the same as defined in the Output Data Structure attribute.

The value of the Output Substitute Data attribute shall be the last valid value of the Output Data element in case of Substitution Mode LAST_VALUE.

8.3.2.2.2.3 Invocation of the Output Data object

For the invocation of the Output Data object the following rules apply:

- Output Data object elements (subslot) data shall not exceed the total length of 1 439 octets. The attributes List of Output Data Elements, Output Data Object Format respectively Format and Length have to be set accordingly.
- The relation of an Output Data object element to a Output CR, Input CR, M CR will be handled in the Context ASE.
- The sum of all Output Data object elements including IOPS and IOCS of the related Input Data object elements related to one CR shall not exceed the total length of 1 440 octets.
- One Output Data object shall be invoked in an application process of an IO device.

8.3.2.3 IO Data service specification

8.3.2.3.1 Set input

The Set Input service sets new input data of one slot/subslot together with its IOPS for next transmission for all related application relationships. The IOPS shall be at any time consistent with delivered Input Data values. The value of IOPS shall only be “GOOD” for valid input data of the whole subslot. In all other cases the value shall be set to “BAD_BY_xxx” and include the detecting instance e.g. subslot, slot or IO Device. This service is used by the application process of the IO device to update the addressed part of the protocol internal cyclic buffer. The actual transmission of the buffer is time triggered.

The Set Input service may set the IOPS only. It is the case if the slot/subslot does not contain Input Data.

NOTE 1 The local implementation of the service interface may support access variants with other than subslot granularity e.g. for more than one subslot or only parts (variables) of the Input Data object element. However, the Input Data object element is always consistent with its IOPS.

This service shall only be used in conjunction with the IO AR and Supervisor AR.

According to the cyclic buffer to buffer transportation characteristics the following behavior is possible:

- The Set Input requests are issued faster than the contents of the buffer is transported over the network. In this case only the values provided with the latest Set Input service are conveyed over the network.
- The Set Input requests are issued slower than the contents of the buffer is transported over the network. In this case the values provided with each Set Input service are conveyed more than once over the network.
- If the Set Input requests are issued synchronous with the transport of the contents of the buffers the values provided with each Set Input service are conveyed once over the network.

Table 247 shows the parameters of the service.

Table 247 – Set input

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOPS	M	
Input Data	U	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

NOTE 2 The Set Input service sets the input data for all established ARs that belong to the submodule. In this case multiple calls for shared inputs are not necessary. Therefore, the AR and CR service parameter are omitted.

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

CREP
This parameter is the local identifier for the desired CR.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

IOPS
This parameter shall contain the values GOOD if data values are valid, BAD_BY_DEVICE, BAD_BY_SLOT, or BAD_BY_SUBSLOT.

Input Data
This parameter contains the value of the Input Data object element.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.2 Set input IOCS

This service shall be used by a consumer (IO controller) of inputs to pass the value of changed IOCS to the application layer. The IOCS is conveyed back to the provider and provides a means to monitor the application. This service shall only be used in conjunction with the IO AR and Supervisor AR.

By this service the application process of the IO device updates the local buffer. According to the cyclic buffer to buffer transportation characteristics of the used application relationship the following behavior is possible:

- The Set Input IOCS requests are issued faster than the contents of the buffers are transported over the network to the Producer. In this case not each value of the IOCS is conveyed over the network. Only the latest value is transported, others are locally overwritten.
- The Set Input IOCS requests are issued slower than the contents of the buffers are transported over the network to the Producer. In this case each value of the IOCS provided with the service is conveyed more than once over the network.
- If the Set Input IOCS requests are issued synchronous with the transport of the contents of the buffers each value of the IOCS provided with the service is conveyed over the network.
- If there are several IOCS than only the IOCS of the non-shared CR will be indicated.

Table 248 shows the parameters of the service.

Table 248 – Set Input IOCS

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOCS	M	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

CREP
This parameter is the local identifier for the desired CR.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

IOCS
This parameter shall contain the values GOOD or BAD_BY_CONTROLLER.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.3 Get Input

This service shall only be used in conjunction with the IO AR and Supervisor AR. It is used by an IO controller to read the values of an input data subslot with the related IOPS.

Table 249 shows the parameters of the service.

Table 249 – Get Input

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOPS		M
Input Data		C
New Flag		M
IOCS		C
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

CREP
This parameter is the local identifier for the desired CR.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

Result(+)
This parameter indicates that the service request succeeded.

IOPS
This parameter contains the value of the corresponding attribute of the Input Data object.

Input Data
This parameter contains the value of the Input Data object.

New Flag
This parameter contains the value TRUE if new Input data with IOPS have been received since last time calling the service. If the service was called more frequently the New Flag is set to FALSE by the protocol machines.

IOCS
This parameter contains the value of the IOCS in case there is corresponding Output Data object.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.4 Get Input IOCS

This service shall only be used in conjunction with the IO AR and Supervisor AR. It is used by a provider to get the IOCS of the IO controller.

Table 250 shows the parameters of the service.

Table 250 – Get Input IOCS

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOCS		M
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

CREP
This parameter is the local identifier for the desired CR.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

Result(+)
This parameter indicates that the service request succeeded.

IOCS
This parameter contains the value of the IOCS.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.5 New Input

The New Input service signals to the application of the IO Controller a reception of a valid APDU. This service shall also be conveyed if the IOPS has been changed. Furthermore, this service is used to indicate that the Watchdog expired with the parameter Watchdog Flag.

Table 251 shows the parameters of the service.

Table 251 – New Input

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Slot Number	M
Subslot Number	M
Watchdog Flag	C
InData Flag	C

Argument
The argument shall convey the service specific parameters of the service indication.

AREP
This parameter is the local identifier for the desired AR passed with the indication.

CREP
This parameter is the local identifier for the desired CR passed with the indication.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

Watchdog Flag
This parameter contains the value WATCHDOG_EXPIRED if no APDU or no valid APDU Data Status has been received within the Watchdog Time Interval. The AR will be terminated in parallel if the Data Hold Time Interval contains the same value. Otherwise the application behavior within the IO controller is a local means.

InData Flag
This parameter indicates that the first valid frame was received. All consumer IOCRs of the AR have got data first time.

8.3.2.3.6 Set Input APDU Data Status

This service shall be used by the IO device to set the flags “Data Flag”, “AR State Flag”, “Provider State Flag”, and “Problem Indicator Flag”. These flags modify the field APDU Data Status of the next conveyed APDU. The flags “Data Valid Flag”, “AR State Flag”, and “Problem Indicator Flag” influence only the selected AR. The flag “Provider State Flag” can influence all established ARs of the IO device.

Table 252 shows the parameters of the service.

Table 252 – Set input APDU data status

Parameter name	Req	Cnf
Argument	M	
AREP	U	
Data Valid Flag	U	
AR State Flag	U	
Provider State Flag	U	
Problem Indicator Flag	U	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service indication.

AREP
This parameter is the local identifier for the desired AR passed with the request. It is omitted if the service parameter Provider State Flag is used because this parameter influences all established ARs transmitting IO Data.

Data Valid Flag
This parameter indicates with the value "DataItem invalid" that all data (inclusive IOPS/IOCS) of the AR are not valid and shall not be processed by the consumer.

NOTE 1 The Data Flag with "DataItem invalid" could be used if the application process of the IO device is not able to set Input Data and/or IOPS/IOCS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS/IOCS should be used to tell the consumer about the status of the input data. This flag is also used for redundancy in future versions.

The value "DataItem valid" shall be the default value for regular operation to indicate that the data values including the IOPS/IOCS are valid.

AR State Flag
This parameter indicates if the AR is operating with the value "primary" or standby with the value "backup".

NOTE The value "backup" is reserved for further versions including system redundancy.

Provider State Flag
This parameter indicates if the application process is running with the value "run" or stopped with the value "stop". This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS.

NOTE This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag
This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE. It is indicated by the value "Problem detected". It shall be set to "Regular operation" as long as no diagnosis entry exists.

NOTE 2 This flag could be used to trigger the reading of diagnosis data of the application within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to "BAD" for local reasons by means of the Set Input service.

The default values for the flags above are:

- Data Flag: "DataItem valid"
- AR State Flag: "primary"
- Provider State Flag: "run"
- Problem Indicator Flag: "Regular operation"

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.7 New Input APDU Data Status

The service signals a changed of the APDU Data Status value to the application of the IO controller. This service shall only be conveyed if the APDU Data Status value has been changed by the IO device and shall provide only the changed flags.

Table 253 shows the parameters of the service.

Table 253 – New Input APDU Data Status

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Data Valid Flag	C
AR State Flag	C
Provider State Flag	C
Problem Indicator Flag	C

Argument
The argument shall convey the service specific parameters of the service indication.

AREP
This parameter is the local identifier for the desired AR passed with the indication.

CREP
This parameter is the local identifier for the desired CR passed with the indication.

Data Valid Flag
This parameter indicates with the value "DataItem invalid" that all data (inclusive IOPS) of the AR are not valid and shall not be processed by the consumer.

NOTE 1 The Data Flag with "DataItem invalid" could be used if the application process of the IO device is not able to set Input Data and/or IOPS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS should be used to tell the consumer about the status of the input data.

The value "DataItem valid" indicate that the data values including the IOPS, IOCS are valid.

AR State Flag
This parameter indicates if the AR is operating with the value "primary" or standby with the value "backup".

NOTE 2 The value "backup" is reserved for further versions including redundancy.

Provider State Flag
This parameter indicates if the application process is running with the value „run" or stopped with the value „stop". This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS.

NOTE 3 This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag
This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE, which affects the related AR. It is indicated by the value „Problem detected". It shall be set to „Regular operation" as long as no diagnosis entry exists.

NOTE 4 This flag could be used to trigger the reading of diagnosis data within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to "BAD" for local reasons by means of the Set Input service.

8.3.2.3.8 Read input data

This confirmed service may be used to read the value of an Input Data object element. This service can be used in conjunction with the IO AR, Supervisor AR, or implicit AR.

NOTE 1 A client application may use the service Read AR Data to read possible AR UUIDs of established ARs that could be used in this context.

Implicit AR using Target AR UUID: This service does only contain an Input Data object if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Implicit AR using API: This service contains an Input Data object.

Table 254 shows the parameters of the service.

Table 254 – Read Input Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Length IOCS			M	M(=)
Length IOPS			M	M(=)
Length Input Data			M	M(=)
IOCS			M	M(=)
IOPS			M	M(=)
Input Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired Input Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired Input Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of a Input Data object that has to be read. The allowed length is from 2^0 to 2^{32}-256. If the requested length exceeds the actual length of the Input Data object element then the actual length with data shall be responded. If the requested length is shorter than the length of the Input Data object element then a parameter error shall be the result.

Result(+)
This parameter indicates that the service request succeeded.

Length IOPS
This parameter contains the value of the corresponding attribute of the Input Data object.

Length IOCS
This parameter contains the value of the corresponding attribute of the Input Data object.

Length Input Data
This parameter contains the data length of the object which has been read.

IOCS
This parameter contains the value of the corresponding attribute of the Input Data object.

IOPS
This parameter contains the value of the corresponding attribute of the Input Data object.

Input Data
This parameter contains the value of the object which has been read and consists of the number of octets indicated in the parameter Length Input Data. This parameter has to be composed of the data types defined in Clause 5.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 may be used by device vendors to transmit specific error messages.

The negative response shall contain the values Error Decode = "PNIORW", Error code 1 = "invalid index", Error code 2 = "user specifc" if the addressed submodule does not contain any input data.

8.3.2.3.9 Set Output

The Set Output service sets new output data of one slot/subslot together with its IOPS for next transmission for all related application relationships. The IOPS shall be at any time consistent with delivered Output Data values. The value of IOPS shall only be "GOOD" for valid output data of the whole subslot. In all other cases the value shall be set to "BAD_BY_CONTROLLER". This service is used by the application process of the IO controller to update the addressed part of the protocol internal cyclic buffer. The actual transmission of the buffer is time triggered.

NOTE The local implementation of the service interface may support access variants with other than subslot granularity e.g. for more than one subslot or only parts (variables) of the Output Data object element. However, the Output Data object element is always consistent with its IOPS.

This service shall only be used in conjunction with the IO AR or Supervisor AR.

According to the cyclic buffer to buffer transportation characteristics the following behavior is possible:

- The Set Output requests issued faster than the contents of the buffers are transported over the network. In this case only the values provided with the latest Set Output service are conveyed over the network.
- The Set Output requests issued slower than the contents of the buffers are transported over the network. In this case the values provided with each Set Output service are conveyed more than once over the network.
- If the Set Output requests are issued synchronous with the transport of the contents of the buffers the values provided with each Set Output service are conveyed once over the network.

Table 255 shows the parameters of the service.

Table 255 – Set Output

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOPS	M	
Output Data	M	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

CREP
This parameter is the local identifier for the desired CR.

Slot Number
This parameter contains the slot number.

Subslot Number
This parameter contains the subslot number.

IOPS
This parameter shall contain the values GOOD if data values are valid or BAD_BY_CONTROLLER if this data are not valid.

Output Data
This parameter contains the value of the Output Data object element.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.10 Set Output IOCS

This service shall be used by a consumer (IO device) of outputs to pass the value of changed IOCS to the application layer. The IOCS is conveyed back to the provider and provides a means to monitor the application. This service shall only be used in conjunction with the IO AR and Supervisor AR.

By this service the application process of the IO device updates the local buffer. According to the cyclic buffer to buffer transportation characteristics of the used application relationship the following behavior is possible:

- The IO Status Change requests issued faster than the contents of the buffers are transported over the network to the Producer. In this case not each value of the IOCS is conveyed over the network. Only the latest value is transported, others are locally overwritten.
- The IO Status Change requests are issued slower than the contents of the buffers are transported over the network to the Producer. In this case each value of the IOCS provided with the service is conveyed more than once over the network.
- If the IO Status Change requests are issued synchronous with the transport of the contents of the buffers each value of the IOCS provided with the service is conveyed over the network.

Table 256 shows the parameters of the service.

Table 256 – Set Output IOCS

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOCS	M	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

CREP
This parameter is the local identifier for the desired CR.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

IOCS
This parameter may contain the values GOOD, BAD_BY_DEVICE, BAD_BY_SLOT, or BAD_BY_SUBSLOT.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.11 Get Output

This service shall be used by an IO device to read the value for one slot/subslot of an Output Data object element together with its IOPS from the application layer. This service shall only be used in conjunction with the IO AR or Supervisor AR.

The IO device shall only process the values of the Output Data object element if the IOPS contains the value GOOD.

Table 257 shows the parameters of the service.

Table 257 – Get Output

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOPS		M
Output Data		M
New Flag		M
IOCS		C
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR passed with the indication.

CREP
This parameter is the local identifier for the desired CR passed with the indication.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

Result(+)
This parameter indicates that the service request succeeded.

IOPS
This parameter contains the value of the corresponding attribute of the Output Data object element.

Output Data
This parameter contains the value of the Output Data object.

New Flag
This parameter contains the value TRUE if new Output data with IOPS have been received since last time calling the service. If the service was called more frequently the New Flag is set to FALSE by the protocol machines.

IOCS
This parameter contains the value of the IOCS in case there is a corresponding Input Data object.

Result(-)

This parameter indicates that the service request failed.

8.3.2.3.12 Get Output IOCS

This service shall be used by an IO device to read the IOCS from the application layer. This service shall only be used in conjunction with the IO AR or Supervisor AR.

Table 258 shows the parameters of the service.

Table 258 – Get Output IOCS

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOCS		M
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR passed with the indication.

CREP
This parameter is the local identifier for the desired CR passed with the indication.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

Result(+)
This parameter indicates that the service request succeeded.

IOCS
This parameter contains the value of the IOCS.

Result(-)

This parameter indicates that the service request failed.

8.3.2.3.13 New Output

The New Output service signals to the application of the IO device a reception of a valid APDU. This service shall also be conveyed if the IOPS has been changed. Furthermore, this service is used to indicate that the Watchdog expired with the parameter Watchdog Flag.

Table 259 shows the parameters of the service.

Table 259 – New Output

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Slot Number	M
Subslot Number	M
Watchdog Flag	C
InData Flag	C

Argument
The argument shall convey the service specific parameters of the service indication.

AREP
This parameter is the local identifier for the desired AR passed with the indication.

CREP
This parameter is the local identifier for the desired CR passed with the indication.

Slot Number
This parameter contains the local slot number.

Subslot Number
This parameter contains the local subslot number.

Watchdog Flag
This parameter contains the value WATCHDOG_EXPIRED if no APDU or no valid APDU Data Status has been received within the Watchdog Time Interval. The AR will be terminated in parallel if the Data Hold Time Interval contains the same value. Otherwise the application behavior within the IO device is to hold the last valid value until the communication continued or the AR is terminated by the Context Management ASE.

InData Flag
This parameter indicates that the first valid frame was received. All consumer IOCRs of the AR have got data first time.

8.3.2.3.14 Set Output APDU Data Status

This service shall be used by the IO controller to set the flags “Data Flag”, “AR State Flag”, “Provider State Flag”, and “Problem Indicator Flag”. These flags modify the field APDU Data Status of the next conveyed APDU. The flags “Data Flag”, “AR State Flag”, and “Problem Indicator Flag” influence only the selected AR. The flag “Provider State Flag” influences all established ARs of the IO device.

Table 260 shows the parameters of the service.

Table 260 – Set Output APDU Data Status

Parameter name	Req	Cnf
Argument	M	
AREP	U	
Data Flag	U	
AR State Flag	U	
Provider State Flag	U	
Problem Indicator Flag	U	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument
The argument shall convey the service specific parameters of the service indication.

AREP
This parameter is the local identifier for the desired AR passed with the request. It is omitted if the service parameter Provider State Flag is used because this parameter influences all established ARs transmitting IO Data.

Data Flag
This parameter indicates with the value “DataItem invalid” that all data (inclusive IOPS) of the AR are not valid and shall not be processed by the consumer.

NOTE 1 The Data Flag with “DataItem invalid” could be used if the application process of the IO controller is not able to set Input Data and/or IOPS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS should be used to tell the consumer about the status of the output data.

The value “DataItem valid” shall be the default value for regular operation to indicate that the data values including the IOPS are valid.

AR State Flag
This parameter indicates if the AR is operating with the value „primary“ or standby with the value „backup“. The default value is „primary“.

NOTE 2 The value „backup“ is reserved for further versions including redundancy.

Provider State Flag
This parameter indicates if the application process is running with the value „run“ or stopped with the value „stop“. This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS. In case of „stop“ all IOPS shall also be set to the value “BAD_BY_CONTROLLER” by means of the Set Input service request to all slots/subslots.

NOTE 3 This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag
This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE, which affects the related AR. It is indicated by the value „Problem detected“. It shall be set to „Regular operation“ as long as no diagnosis entry exists. It is also the default value.

NOTE 4 This flag could be used to trigger the reading of diagnosis data within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to “BAD” for local reasons by means of the Set Input service.

In summary, initial or default values for the flags above are:

- Data Flag: “DataItem valid”
- AR State Flag: „primary“
- Provider State Flag: „run“
- Problem Indicator Flag: „Regular operation“

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

8.3.2.3.15 New Output APDU Data Status

The service signals a changed of the APDU Data Status value to the application of the IO device. This service shall only be conveyed if the APDU Data Status value has been changed by the IO controller and shall provide only the changed flags.

Table 261 shows the parameters of the service.

Table 261 – New Output APDU Data Status

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Data Flag	C
AR State Flag	C
Provider State Flag	C
Problem Indicator Flag	C

Argument
The argument shall convey the service specific parameters of the service indication.

AREP
This parameter is the local identifier for the desired AR passed with the indication.

CREP
This parameter is the local identifier for the desired CR passed with the indication.

Data Flag
This parameter indicates with the value DATA_INVALID that all data (inclusive IOPS) of the AR are not valid and shall not be processed by the consumer.

NOTE 1 The Data Flag with DATA_INVALID could be used if the application process of the IO controller is not able to set Input Data and/or IOPS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS should be used to tell the consumer about the status of the input data. This flag is also used for redundancy in future versions.

The value “DataItem valid” shall be the default value for regular operation to indicate that the data values including the IOPS are valid.

AR State Flag
This parameter indicates if the AR is operating with the value „primary“ or standby with the value „backup“. The default value is „primary“.

NOTE The value „backup“ is reserved for further versions including redundancy.

Provider State Flag
This parameter indicates if the application process is running with the value „run“ or stopped with the value „stop“. This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS. In case of „stop“ all IOPS shall also be set to the value “BAD_BY_CONTROLLER” by means of the Set Input service request to all slots/subslots.

NOTE This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag
This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE. It is indicated by the value „Problem detected“. It shall be set to „Regular operation“ as long as no diagnosis entry exists. It is also the default value.

NOTE 2 This flag could be used to trigger the reading of diagnosis data within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to “BAD” for local reasons by means of the Set Input service.

8.3.2.3.16 Read Output Data

This confirmed service may be used to read the value of an Output Data object element with the substitute value. This service can be used in conjunction with the IO AR, Supervisor AR, or the implicit AR.

NOTE 1 A client application may use the service Read AR Data to read possible AR UUIDs of established ARs that could be used in this context.

Implicit AR using Target AR UUID: This service does only contain an Output Data object if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Implicit AR using API: This service does only contain an Output Data object if the application supports reading of values of Output Data objects without an established AR. Otherwise a parameter error is responded.

Table 262 shows the parameters of the service.

Table 262 – Read Output Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Length IOCS			M	M(=)
Length IOPS			M	M(=)
Length Output Data			M	M(=)
IOCS			M	M(=)
IOPS			M	M(=)
Output Data			M	M(=)
Substitute Mode			M	M(=)
Substitute Active Flag			M	M(=)
Output Substitute Data Valid			M	M(=)
Output Substitute Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired Output Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired Output Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16}-1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of a Output Data object that has to be read. The allowed length is from 2^{0} to 2^{32}-256. If the requested length exceeds the actual length of the Output Data object element then the actual length with data shall be responded. If the requested length is shorter than the length of the Output Data object element then a parameter error shall be the result.

Result(+)
This parameter indicates that the service request succeeded.

Length IOPS
This parameter contains the value of the corresponding attribute of the Output Data object.

Length IOCS
This parameter contains the value of the corresponding attribute of the Output Data object.

Length Output Data
This parameter contains the data length of the object which has been read.

IOCS
This parameter contains the value of the corresponding attribute of the Output Data object.

IOPS
This parameter contains the value of the corresponding attribute of the Output Data object.

Output Data
This parameter contains the value of the object which has been read and consists of the number of octets indicated in the Length Output Data of the response. This parameter has to be composed of the data types defined in Clause 5.

Substitute Mode
This parameter contains the value of the corresponding attribute of the Output Data object.

Substitute Active Flag
This parameter contains the value TRUE if substitute data are used in case of a failure.

Output Substitute Data Valid
This parameter contains the value of the corresponding attribute of the Output Data object.

Output Substitute Data
This parameter contains the substitute value of the object which has been read and consists of the number of octets indicated in the Length Output Data of the response. This parameter has to be composed of the data types defined in Clause 5.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 2 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 3 Add Data 2 may be used by device vendors to transmit specific error messages.

The negative response shall contain the values Error Decode = "PNIORW", Error code 1 = "invalid index", Error code 2 = "user specifc" if the addressed submodule does not contain any output data.

8.3.2.3.17 Read Output Substitute Data

This confirmed service may be used to read the substitute value. This service can be used in conjunction with the IO AR, Supervisor AR, or the implicit AR.

NOTE 1 A client application may use the service Read AR Data to read possible AR UUIDs of established ARs that could be used in this context.

Implicit AR using Target AR UUID: This service does only contain an Output Data object if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Implicit AR using API: This service does only contain an Output Data object if the application supports reading of values of Output Data objects without an established AR. Otherwise a parameter error is responded.

Table 263 shows the parameters of the service.

Table 263 – Read Output Substitute Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Length Output Data			M	M(=)
Substitute Mode			M	M(=)
Substitute Active Flag			M	M(=)
Output Substitute Data Valid			M	M(=)
Output Substitute Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired Output Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired Output Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of a Output Data object that has to be read. The allowed length is from 2^0 to 2^{32}-256. If the requested length exceeds the actual length of the Output Data object element then the actual length with data shall be

responded. If the requested length is shorter than the length of the Output Data object element then a parameter error shall be the result.

Result(+)
This parameter indicates that the service request succeeded.

Length Output Data
This parameter contains the data length of the object which has been read.

Substitute Mode
This parameter contains the value of the corresponding attribute of the Output Data object.

Substitute Active Flag
This parameter contains the value TRUE if substitute data are used in case of a failure.

Output Substitute Data Valid
This parameter contains the value of the corresponding attribute of the Output Data object.

Output Substitute Data
This parameter contains the substitute value of the object which has been read and consists of the number of octets indicated in the Length Output Data of the response. This parameter has to be composed of the data types defined in Clause 5.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 may be used by device vendors to transmit specific error messages.

The negative response shall contain the values Error Decode = "PNIORW", Error code 1 = "invalid index", Error code 2 = "user specifc" if the addressed submodule does not contain any output data.

8.3.2.3.18 Write Output Substitute Data

This confirmed service may be used to write the substitute value of an Output Data object element. This service can be used in conjunction with the IO AR and Supervisor AR. Table 264 shows the parameters of the service.

The service parameter “Multiple” may be used to convey substitute values within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

Table 264 – Write Output Substitute Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Substitute Mode	M	M(=)		
Length Output Data	M	M(=)		
Output Substitute Data	M	M(=)		
Prm Flag	M			
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired Output Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired Output Data object in a specific subslot (typically a submodule).

Multiple
The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Substitute Mode
This parameter contains the value of the corresponding attribute of the Output Data object.

Length Output Data
This parameter contains the value of the corresponding attribute of the Output Data object.

Output Substitute Data
This parameter contains the substitute value of the object which shall be written and consists of the number of octets indicated in the parameter Length Output Data. This parameter has to be composed of the data types defined in 0. In case of Substitute Mode ZERO and LAST_VALUE this parameter shall contain the value zero for each data octet. In case of Substitute Mode PROFILE_SPECIFIC the content of this parameter shall be defined by the profile.

Prm Flag
The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

NOTE 1 This parameter is used to control the persistent storage of data. It is utilized in fast start-up procedure.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 2 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 3 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.2.4 Behavior of IO Data objects

8.3.2.4.1 General behavior of the Input Data object

For performance reasons, the value of the Input Data object shall be transferred on change as fast as possible to the IO AR buffer for transmission. The speed and the jitters of this action is a performance parameter of an IO device.

8.3.2.4.2 General behavior of the Output Data object

For performance reasons, the value for the Output Data object shall be transferred on receipt as fast as possible to the object. The speed and the jitters of this action is a performance parameter of a IO device. This parameter determines the synchronization accuracy.

The Read Output service shall provide the current value of the application process which has been fetched with the Get Output service.

NOTE Due to different reasons (e.g. physical output delays) this value may differ from the value of the real output data or from the last transferred value.

8.3.3 Logbook Data ASE

8.3.3.1 Overview

The Logbook Data ASE provides a mean to store locally generated and AR specific events in a circular buffer and provides a service to read that event buffer. The content of entries is manufacturer specific and may be used to identify communication problems e.g. reasons for AR termination.

The Logbook Data ASE defines attributes of Logbook Data objects and provides a set of services used to read remotely, and write their values locally. A Logbook Data ASE is specific for an IO Device. The Logbook Data object is addressed implicitly by the defined services. The entries shall always be responded from the newest to the oldest one up to the requested length.

The access to Logbook Data objects is performed according to the Client/Server access model for read. The Client/Server model is characterized by a Client application sending a read request to a Server application that responds accordingly.

The formal model of the Logbook Data ASE is presented by the Logbook Data class specification, containing a description of its attributes, services as well as invocations, followed by a detailed service specification.

8.3.3.2 Logbook Data class specification

8.3.3.2.1 Template

A Logbook Data object is described by the following template:

ASE:	**Logbook Data ASE**		
CLASS:	**Logbook Data**		
CLASS ID:	**not used**		
PARENT CLASS:	**TOP**		
ATTRIBUTES:			
1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	Current Local Time Stamp
3	(m)	Attribute:	List of Entries

3.1	(m)	Attribute:	Local Time Stamp
3.2	(m)	Attribute:	AR UUID
3.3	(m)	Attribute:	PNIO Status
3.4	(m)	Attribute:	Entry Detail

SERVICES:

1	(m)	OpsService:	Read Logbook
2	(m)	OpsService:	Logbook Event

8.3.3.2.2 Attributes

Implicit
The attribute Implicit indicates that the Logbook Data object is implicitly addressed by the services.

Current Local Time Stamp
This attribute contains the current local time by means of the value of the cycle counter.

Attribute type: Unsigned64

List of Entries
An IO Device shall support at least 16 logbook entries and an IO Controller shall support at least 4 KByte. This attribute list consists of the following attributes.

Local Time Stamp
This attribute contains the local time when the entry was made by means of the value of the cycle counter at this point of time.

Attribute type: Unsigned64

AR UUID
This attribute defines the UUID of the IO AR entry which is the source.

Attribute type: UUID

PNIO Status
This attribute contains the protocol machine defined value of the entry.

Attribute type: Unsigned32

Allowed Values: see Part 6 of this series of International Standards

Entry Detail
This attribute contains the protocol machine defined value of the entry.

Attribute type: Unsigned32

8.3.3.3 Logbook Data service specification

8.3.3.3.1 Read logbook

This confirmed service may be used to read the value of a Logbook Data object. This service can be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 265 shows the parameters of the service.

Table 265 – Read Logbook

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Current Local Time Stamp			M	M(=)
List of Entries			M	M(=)
Local Time Stamp			M	M(=)
AR UUID			M	M(=)
PNIO Status			M	M(=)
Entry Detail			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of a Logbook Data object that has to be read. The allowed length is from 0 to 2^{32}-65.

Result(+)
This parameter indicates that the service request succeeded.

Current Local Time Stamp
This parameter contains the value of the corresponding attribute of the ASE object.

List of Entries
This parameter is composed of the following list elements:

NOTE 1 If list of entries have more bytes than the requested length, only the newest entries should used for the response.

Local Time Stamp
This parameter contains the value of the corresponding attribute of the ASE object.

AR UUID
This parameter contains the value of the corresponding attribute of the ASE object.

PNIO Status
This parameter contains the value of the corresponding attribute of the ASE object.

Entry Detail
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 2 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 3 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.3.3.2 Logbook Event

This local service indicates value for the Logbook Data object. Table 266 shows the parameters of the service.

Table 266 – Logbook Event

Parameter name	Ind
Argument	M
AREP	M
Local Time Stamp	M
AR UUID	M
PNIO Status	M
Entry Detail	M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Local Time Stamp
This parameter contains the value of the corresponding attribute of the ASE object.

AR UUID
This parameter contains the value of the corresponding attribute of the ASE object.

PNIO Status
This parameter contains the value of the corresponding attribute of the ASE object.

Entry Detail
This parameter contains the value of the corresponding attribute of the ASE object.

8.3.4 Diagnosis ASE

8.3.4.1 Overview

In the application layer environment, the Diagnosis ASE of an IO device contains different Diagnosis entries that are used to keep the diagnosis information of the IO device. The diagnosis information from the IO device is related to the addressed AP and includes only those slots, subslots and channels which have set diagnosis.

NOTE If a subslot or channel does not have diagnosis set then this information it is not transmitted.

The Diagnosis ASE defines attributes of the Diagnosis objects and provides a set of services used to read their values. The following Diagnosis types are defined:

- Channel Diagnosis
  - Diagnosis
  - Maintenance (required and demanded)
- Ext Channel Diagnosis
  - Diagnosis
  - Maintenance (required and demanded)
- Qualified Ext Channel Diagnosis
  - Diagnosis
  - Maintenance (required and demanded)
  - Qualified data (Qualifier_2 to Qualifier_31)
- Manufacturer Specific Diagnosis:
  - Manufacturer Specific Diagnosis indicating a failure (Channel Properties Specifier value: Appears)
  - Manufacturer Specific Diagnosis indicating a status (Channel Properties Specifier value: Disappears)
  - Maintenance (required and demanded)

It is strongly recommended to use Channel Diagnosis, Ext Channel Diagnosis, or Qualified Ext Channel Diagnosis in all matching cases because detailed semantics are fully standardized. Additional vendor specific diagnosis data can be conveyed by means of the Manufacturer Specific Diagnosis.

The application of an IO device may provide the value of one or several Diagnosis objects to the implicit AR, IO AR, and the Supervisor AR.

The Diagnosis model uses the Client/Server access model.

Furthermore, in case of setting the first diagnosis entry the Input Data ASE shall be informed by means of the event (Diagnosis ASE "First Diagnosis Entry set"). In case of unsetting the last diagnosis entry the Input Data ASE shall be informed by means of the event (Diagnosis ASE "Last Diagnosis Entry cleared"). The Input Data ASE shall not be informed if only maintenance information is set or cleared.

The Alarm ASE may also be informed to send a Diagnosis Alarm, Redundancy Alarm, Diagnosis Disappears Alarm, Multicast Communication Mismatch Alarm, Port Data Change

Notification Alarm, Sync Data Change Notification Alarm, and Isochronous Mode Problem Notification Alarm if it is configured by the application.

8.3.4.2 Diagnosis class specification

8.3.4.2.1 Template

The Diagnosis object is described by the following template:

ASE: **Diagnosis ASE**
CLASS: **Diagnosis**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	Device Status
3	(c)	Constraint:	Device Status <> DEVICE_OK
3.1	(m)	Attribute:	List of APs
3.1.1	(m)	Attribute:	API
3.1.2	(m)	Attribute:	List of Slots
3.1.2.1	(m)	Attribute:	Slot Number
3.1.2.2	(m)	Attribute:	List of Subslots
3.1.2.2.1	(m)	Attribute:	Subslot Number
3.1.2.2.2	(m)	Attribute:	Submodule Diagnosis State
3.1.2.2.3	(m)	Attribute:	List of Channels
3.1.2.2.3.1	(m)	Attribute:	Channel Number
3.1.2.2.3.2	(o)	Attribute:	List of Channel Diagnosis
3.1.2.2.3.2.1	(m)	Attribute:	Channel Properties
3.1.2.2.3.2.1.1	(m)	Attribute:	Type
3.1.2.2.3.2.1.2	(m)	Attribute:	Accumulative
3.1.2.2.3.2.1.3	(m)	Attribute:	Maintenance Required
3.1.2.2.3.2.1.4	(m)	Attribute:	Maintenance Demanded
3.1.2.2.3.2.1.5	(m)	Attribute:	Specifier
3.1.2.2.3.2.1.6	(m)	Attribute:	Direction
3.1.2.2.3.2.2	(m)	Attribute:	Channel Error type
3.1.2.2.3.3	(o)	Attribute:	List of Ext Channel Diagnosis
3.1.2.2.3.3.1	(m)	Attribute:	Channel Properties
3.1.2.2.3.3.1.1	(m)	Attribute:	Type
3.1.2.2.3.3.1.2	(m)	Attribute:	Accumulative
3.1.2.2.3.3.1.3	(m)	Attribute:	Maintenance Required
3.1.2.2.3.3.1.4	(m)	Attribute:	Maintenance Demanded
3.1.2.2.3.3.1.5	(m)	Attribute:	Specifier
3.1.2.2.3.3.1.6	(m)	Attribute:	Direction
3.1.2.2.3.3.2	(m)	Attribute:	Channel Error type
3.1.2.2.3.3.3	(m)	Attribute:	Ext Channel Error type
3.1.2.2.3.3.4	(m)	Attribute:	Ext Channel Add Value
3.1.2.2.3.4	(o)	Attribute:	List of Qualified Ext Channel Diagnosis
3.1.2.2.3.4.1	(m)	Attribute:	Channel Properties
3.1.2.2.3.4.1.1	(m)	Attribute:	Type
3.1.2.2.3.4.1.2	(m)	Attribute:	Accumulative
3.1.2.2.3.4.1.3	(m)	Attribute:	Maintenance Required
3.1.2.2.3.4.1.4	(m)	Attribute:	Maintenance Demanded
3.1.2.2.3.4.1.5	(m)	Attribute:	Specifier
3.1.2.2.3.4.1.6	(m)	Attribute:	Direction

3.1.2.2.3.4.2	(m)	Attribute:	Channel Error type
3.1.2.2.3.4.3	(m)	Attribute:	Ext Channel Error type
3.1.2.2.3.4.4	(m)	Attribute:	Ext Channel Add Value
3.1.2.2.3.4.5	(m)	Attribute:	Qualified Channel Qualifier
3.1.2.2.3.5	(o)	Attribute:	List of Manufacturer Specific Diagnosis
3.1.2.2.3.5.1	(m)	Attribute:	Channel Properties
3.1.2.2.3.5.1.1	(m)	Attribute:	Type
3.1.2.2.3.5.1.2	(m)	Attribute:	Accumulative
3.1.2.2.3.5.1.3	(m)	Attribute:	Maintenance Required
3.1.2.2.3.5.1.4	(m)	Attribute:	Maintenance Demanded
3.1.2.2.3.5.1.5	(m)	Attribute:	Specifier
3.1.2.2.3.5.1.6	(m)	Attribute:	Direction
3.1.2.2.3.5.2	(m)	Attribute:	User Structure Identifier
3.1.2.2.3.5.3	(m)	Attribute:	List of Data
3.1.2.2.3.5.3.1	(m)	Attribute:	Data

SERVICES:

1	(m)	OpsService:	Read Device Diagnosis
2	(m)	OpsService:	Diagnosis Event

8.3.4.2.2 Attributes

Implicit
The attribute Implicit indicates that the Diagnosis object is implicitly addressed by the services.

Device Status
This attribute contains the diagnosis summary. If the value is set to DEVICE_OK no slot/subslot shall contain diagnosis information. If the value is set to DEVICE_FAILURE, MAINTENANCE, or MANUFACTURER_SPECIFIC_STATUS at least one slot/subslot shall contain diagnosis information, mainenance or manufacturer specific status information.

Attribute type: Unsigned8

Allowed values: DEVICE_OK, DEVICE_FAILURE, MAINTENANCE or MANUFACTURER_SPECIFIC_STATUS

List of APs
This attribute contains the diagnosis, maintenance, or manufacturer specific status information specific for each supported application process. A list element consists of the following attributes.

API
This attribute shall consist only of APIs with diagnosis maintenance, or manufacturer specific status information.

List of Slots
This attribute contains only slots with diagnosis maintenance, or manufacturer specific status information. A list element consists of the following attributes.

Slot Number
This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Subslots
This attribute contains only subslots with diagnosis maintenance, or manufacturer specific status information. A list element consists of the following attributes.

Subslot Number
This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending

order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE 1 Subslot Number 0 is used in conjunction with AlarmType "Pull" to address a certain module within the IO device; it never contains a diagnosis.

Submodule Diagnosis State
This attribute contains the diagnosis summary for the submodule. The value SUBMODULE_FAILURE shall only be set in case of at least on Channel Diagnosis severity other than INFORMATION and at least one Manufacturer Specific Diagnosis with severity other than INFORMATION and channel properties specifier shall be set to Appears. In all other cases the value shall be set to SUBMODULE_OK.

Attribute type: Unsigned8

Allowed values: SUBMODUL_OK, SUBMODUL_FAILURE

List of Channels
This attribute contains only channels with diagnosis or maintenance information. A list element consists of the following attributes.

Channel Number
This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

In these cases, the attribute Channel Properties Accumulative shall contain the value INDIVIDUAL.

If the attribute Channel Properties Accumulative contains the value CHANNELGROUP then this attribute contains the number of the lowest effected channel.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF user specific channel number, 0x8000 submodule

List of Channel Diagnosis
This attribute contains a list of channel diagnosis. A list element consists of the following attributes.

Channel Properties
This attribute consists of the following attributes. The dependencies according to Table 267 shall be considered.

Table 267 – Dependencies within channel properties

Maintenance Required	Maintenance Demanded	Specifier	Meaning
MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	APPEARS	Maintenance is required
NO_MAINTENANCE_REQUIRED	MAINTENANCE_DEMANDED	APPEARS	Maintenance is demanded
NO_MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	APPEARS	Channel Diagnosis
MAINTENANCE_REQUIRED	MAINTENANCE_DEMANDED	APPEARS	Qualified Channel Diagnosis

Type
This attribute indicates the channel-size to which the Channel Diagnosis object is related.

Allowed values:

UNSPECIFIC shall be used if the Channel Number contains the value 0x8000 or no other matches,

1BIT, 2BIT, 4BIT, BYTE, WORD, 2WORDS, 4WORDS

Accumulative
This attribute indicates if the diagnosis is related to a certain channel or to a group of channels.

Allowed values: INDIVIDUAL, CHANNELGROUP

Maintenance Required
This attribute indicates that a user activity for the channel is recommended in order to keep the application running.

NOTE 2 For example, a printer indicates “toner low” as a maintenance requirement e.g. to order a new toner cartridge.

Allowed values: NO_MAINTENANCE_REQUIRED, MAINTENANCE_REQUIRED

Maintenance Demanded
This attribute indicates that a user activity for the channel is highly recommended in order to keep the application running.

NOTE 3 For example, a printer indicates “toner low, print quality affected” as a maintenance demand to replace the toner cartridge.

Allowed values: NO_MAINTENANCE_DEMANDED, MAINTENANCE_DEMANDED

Specifier
This attribute indicates that the Channel Diagnosis or Maintenance appears.

Allowed values: APPEARS_ means channel diagnosis, maintenance required or maintenance demanded

NOTE 4 For example, a printer indicates “toner empty, printer stopped” as a diagnosis appears.

Direction
This attribute indicates the direction of the channel to which the Channel Diagnosis object is related.

Attribute type: Unsigned8

Allowed values: MANUFACTURER_SPECIFIC, INPUT_CHANNEL, OUTPUT_CHANNEL, BIDIRECTIONAL_CHANNEL

Channel Error type
This attribute indicates the error type of the Channel Related Diagnosis.

Attribute type: Unsigned16

Allowed values:

SHORT_CIRCUIT,

UNDERVOLTAGE,

OVERVOLTAGE,

OVERLOAD,

OVERTEMPERATURE,

LINE_BREAK,

UPPER_LIMIT_VALUE_EXCEEDED,

LOWER_LIMIT_VALUE_EXCEEDED,

ERROR,

SIMULATION_ACTIVE,

MANUFACTURER_SPECIFIC_1 recommended for “parametrization fault”,

MANUFACTURER_SPECIFIC_2 recommended for “power supply fault”,

MANUFACTURER_SPECIFIC_3 recommended for “fuse blown / open”,

MANUFACTURER_SPECIFIC_4,

MANUFACTURER_SPECIFIC_5 recommended for “ground fault”,

MANUFACTURER_SPECIFIC_6 recommended for “reference point lost”,

MANUFACTURER_SPECIFIC_7 recommended for “process event lost / sampling error”,

MANUFACTURER_SPECIFIC_8 recommended for “threshold warning”,

MANUFACTURER_SPECIFIC_9 recommended for “output disabled”,

MANUFACTURER_SPECIFIC_10 recommended for “safety event”,

MANUFACTURER_SPECIFIC_11 recommended for “external fault”,

MANUFACTURER_SPECIFIC_12 up to MANUFACTURER_SPECIFIC_001F

MANUFACTURER_SPECIFIC_0100 up to MANUFACTURER_SPECIFIC_7FFF

List of Ext Channel Diagnosis
This attribute contains a list of extended channel diagnosis or maintenance. A list element consists of the following attributes.

Channel Properties
This attribute has the same definition as in List of Channel Diagnosis.

Channel Error type
This attribute indicates the error type of the Channel Related Diagnosis.

Attribute type: Unsigned16

Allowed values: same values as specified for List of Channel Diagnosis. Furthermore, the following additional values are allowed.

DATA_TRANSMISSION_IMPOSSIBLE,

REMOTE_MISMATCH,

MEDIA_REDUNDANCY_MISMATCH,

SYNC_MISMATCH,

ISOCHRONOUS_MODE_MISMATCH,

MULTICAST_CR_MISMATCH,

FIBER_OPTIC_MISMATCH,

NETWORK_COMPONENT_FUNCTION_MISMATCH

Ext Channel Error type
This attribute depends on the attribute Channel Error type and indicates an extended error type. The allowed values are shown in Table 268.

Attribute type: Unsigned16

Table 268 – Ext Channel Error type

ChannelErrorType	Value
SHORT_CIRCUIT	manufacturer specific, ACCUMULATIVE_INFO
UNDERVOLTAGE	manufacturer specific, ACCUMULATIVE_INFO
OVERVOLTAGE	manufacturer specific, ACCUMULATIVE_INFO
OVERLOAD	manufacturer specific, ACCUMULATIVE_INFO
OVERTEMPERATURE	manufacturer specific, ACCUMULATIVE_INFO
LINE_BREAK	manufacturer specific, ACCUMULATIVE_INFO
UPPER_LIMIT_VALUE_EXCEEDED	manufacturer specific, ACCUMULATIVE_INFO
LOWER_LIMIT_VALUE_EXCEEDED	manufacturer specific, ACCUMULATIVE_INFO
ERROR	manufacturer specific, ACCUMULATIVE_INFO
PROFILE_SPECIFIC_0020 up to PROFILE_SPECIFIC_00FF	common profile specific, ACCUMULATIVE_INFO
MANUFACTURER_SPECIFIC_1 up to MANUFACTURER_SPECIFIC_001F	manufacturer specific, ACCUMULATIVE_INFO
MANUFACTURER_SPECIFIC_0100 up to MANUFACTURER_SPECIFIC_7FFF	manufacturer specific, ACCUMULATIVE_INFO
DATA_TRANSMISSION_IMPOSSIBLE	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	PORT_STATE_MISMATCH_LINK_DOWN
	MAU_TYPE_MISMATCH
	LINE_DELAY_MISMATCH
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF
REMOTE_MISMATCH	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	PEER_CHASSIS_ID_MISMATCH
	PEER_PORT_ID_MISMATCH
	PT_CLASS_3_MISMATCH
	PEER_MAUTYPE_MISMATCH
	PEER_MRP_DOMAIN_MISMATCH
	NO_PEER_DETECTED
	PEER_MRRT_MISMATCH
	PEER_CABLEDELAY_MISMATCH
	PEER_PTCP_MISMATCH
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF
MEDIA_REDUNDANCY_MISMATCH	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	MANAGER_ROLE_FAIL
	MRP_RING_OPEN
	MRRT_RING_OPEN
	MULTIPLE_MANAGER
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF
SYNC_MISMATCH	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	NO_SYNC_MESSAGE_RECEIVED
	WRONG_PTCP_SUBDOMAIN_ID
	WRONG_IR_DATA_ID
	JITTER_OUT_OF_BOUNDARY
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF

ChannelErrorType	Value
ISOCHRONOUS_MODE_MISMATCH	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	OUTPUT_TIME_FAILURE
	INPUT_TIME_FAILURE
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF
MULTICAST_CR_MISMATCH	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	MULTICAST_CONSUMER_CR_TIMED_OUT
	ADDRESS_RESOLUTION_FAILED
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF
FIBRE_OPTIC_MISMATCH	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	POWER_BUDGET
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF
NETWORK_COMPONENT_FUNCTION_MISMATCH	MANUFACTURER_SPECIFIC_0001 up to MANUFACTURER_SPECIFIC_7FFF
	FRAME_DROPPED_NO_RESOURCES
	PROFILE_SPECIFIC_9000 up to PROFILE_SPECIFIC_9FFF

Ext Channel Add Value
This attribute contains an additional value. In case the attribute Ext Channel Error type contains the value ACCUMULATIVE_INFO then the value of the attribute shall have the meaning according to Table 269.

Table 269 – Ext Channel Add Value for Accumulative Info

Value	Meaning
Bit0: 0x00	ChannelNumber is not effected
Bit0: 0x01	ChannelNumber is effected
Bit1: 0x00	ChannelNumber +1 is not effected
Bit1: 0x01	ChannelNumber +1 is effected
Bit1: 0x00	ChannelNumber +1 is not effected
Bit1: 0x01	ChannelNumber +1 is effected
Bit2: 0x00	ChannelNumber +2 is not effected
Bit2: 0x01	ChannelNumber +2 is effected
...	...
Bit30: 0x00	ChannelNumber +30 is not effected
Bit30: 0x01	ChannelNumber +30 is effected
Bit31: 0x00	ChannelNumber +31 is not effected
Bit31: 0x01	ChannelNumber +31 is effected

Attribute type: Unsigned32

List of Manufacturer Specific Diagnosis
This attribute contains a list of manufacturer specific diagnosis, maintenance, or manufacturer specific status information. A list element consists of the following attributes.

Channel Properties
This attribute has the same definition as in the attribute List of Channel Diagnosis. However, the combination of attribute values is extended in order to indicate status information. The dependencies according to Table 270 shall be considered.

Table 270 – Dependencies within Channel Properties for manufacturer specific diagnosis

Maintenance Required	Maintenance Demanded	Specifier	Meaning
MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	APPEARS	Maintenance is required
NO_MAINTENANCE_REQUIRED	MAINTENANCE_DEMANDED	APPEARS	Maintenance is demanded
NO_MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	APPEARS	Manufacturer Diagnosis
NO_MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	DISAPPEARS	Manufacturer Status

Specifier
This attribute indicates that Manufacturer Specific Diagnosis or Maintenance appears, or Manufacturer Specific Status appears.

Allowed values: APPEARS – means manufacturer specific diagnosis, maintenance required or maintenance demanded
DISAPPEARS – means manufacturer specific status appears

User Structure Identifier
This attribute indicates the structure of the Manufacturer Specific Diagnosis Data attribute.

Attribute type: Unsigned16

Allowed values:
MANUFACTURER_SPECIFIC_1 up to MANUFACTURER_SPECIFIC_0x7FFF,
CHANNEL_DIAGNOSIS,
EXT_CHANNEL_DIAGNOSIS,
QUALIFIED_CHANNEL_DIAGNOSIS,
MAINTENANCE,
PROFILE_SPECIFIC
MULTIPLE_DIAGNOSIS

List of Data
This attribute contains a list of manufacturer specific diagnosis data. A list element consists of the following attributes.

Data
This attribute contains manufacturer specific diagnosis data.

Attribute type: One of the following data types shall be used:
Boolean, Binary Date, Integer, Time of Day, Unsigned, Time-Difference, Floating Point, Network Time, VisibleString, Network Time Difference, OctetString

List of Qualified Ext Channel Diagnosis
This attribute contains a list of qualified extended channel diagnosis or maintenance. A list element consists of the following attributes.

Channel Properties
This attribute has the same definition as in List of Ext Channel Diagnosis.

Channel Error type
This attribute has the same definition as in List of Ext Channel Diagnosis.

Ext Channel Error type
This attribute has the same definition as in List of Ext Channel Diagnosis.

Ext Channel Add Value
This attribute has the same definition as in List of Ext Channel Diagnosis.

Qualified Channel Qualifier
This attribute has shall contain an additional qualifier for diagnosis data. The values shall be used profile specific.

Attribute type: Unsigned32

Allowed Values: QUALIFIER_2_NOTSET, QUALIFIER_2_SET, ..., QUALIFIER_31_NOTSET, QUALIFIER_31_SET

8.3.4.2.3 Invocation of the Diagnosis object

An IO device shall invoke one Diagnosis object for each application process which shall contain the diagnosis information of the IO device with its faulty or maintenance information containing slots, subslots and channels.

8.3.4.3 Diagnosis service specification

8.3.4.3.1 Read Device Diagnosis

This confirmed service may be used to read the value of an AP Device Diagnosis object. This service shall only be used in conjunction with the implicit AR (Target AR UUID or API shall be set), IO AR or Supervisor AR.

By means of the service parameter API, AR UUID, Slot, and Subslot the content of diagnosis can be restricted to a specific API, AR, a specific Slot or a specific Subslot as a filter function. In case a user specific filter is selected the Read Diagnosis response shall only contain diagnosis items that match the filter criteria and contain diagnosis, maintenance or manufacturer specific status information. Otherwise all items with diagnosis, maintenance and manufacturer specific status information shall be responded.

Table 271 shows the parameters of the service.

Table 271 – Read Device Diagnosis

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Diagnosis Item	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of Diagnosis Data			M	M(=)
Channel Diagnosis			S	S(=)
Block Version Low = 0			S	S(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Channel Diagnostic Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error type			M	M(=)
Block Version High = 1			S	S(=)
API			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Channel Diagnostic Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error type			M	M(=)
Ext Channel Diagnosis			S	S(=)

Parameter name	Req	Ind	Rsp	Cnf
Block Version Low = 0			S	S(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Ext Channel Diagnosis Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error type			M	M(=)
Ext Channel Error type			M	M(=)
Ext Channel Add Value			M	M(=)
Block Version Low = 1			S	S(=)
API			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Ext Channel Diagnosis Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error type			M	M(=)
Ext Channel Error type			M	M(=)
Ext Channel Add Value			M	M(=)
Qualified Ext Channel Diagnosis			S	S(=)
Block Version Low = 0			S	S(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Qualified Ext Channel Diagnosis Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error type			M	M(=)
Ext Channel Error type			M	M(=)
Qualified Channel Qualifier			M	M(=)
Block Version Low = 1			S	S(=)
API			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Qualified Ext Channel Diagnosis Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error type			M	M(=)
Ext Channel Error type			M	M(=)
Ext Channel Add Value			M	M(=)
Qualified Channel Qualifier			M	M(=)
Manufacturer Specific Diagnosis			S	S
Block Version Low = 0			S	S(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
User Structure Identifier			M	M(=)
List of Data			M	M(=)
Data			M	M(=)
Block Version Low = 1			S	S(=)
API			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
User Structure Identifier			M	M(=)
List of Data			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall only be used to read AR specific diagnosis information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE It is used to read only diagnosis information the requested AR is connected to.

Slot Number
The parameter Slot Number is used to address only diagnosis information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used is used to address only diagnosis information for a specific slot/subslot. This service parameter shall only be used together with the service parameter Slot Number.

8.3.4.3.2 Diagnosis Item

This parameter is used to select the type of diagnosis and/or maintenance information. The allowed values shall be according Table 272.

Table 272 – Diagnosis Item

Diagnosis Item Value	Query
ALL	Channel Diagnosis, ExtChannel Diagnosis, Manufacturer Specific Diagnosis with status, Maintenance
CHANNEL	Channel Diagnosis, ExtChannel Diagnosis
CHANNEL_AND_MANUFACTURER	Channel Diagnosis, ExtChannel Diagnosis, Manufacturer Specific Diagnosis
CHANNEL_MAINTENANCE_REQUIRED	Maintenance Required information associated with Channel Diagnosis, Maintenance Required information associated with ExtChannel Diagnosis
CHANNEL_MAINTENANCE_DEMANDED	Maintenance Demanded information associated with Channel Diagnosis, Maintenance Demanded information associated with ExtChannel Diagnosis
CHANNEL_MANUFACTURER_MAINTENANCE_REQUIRED	Maintenance Required information associated with Channel Diagnosis, Maintenance Required information associated with ExtChannel Diagnosis, Maintenance Required information associated with Manufacturer Specific Diagnosis
CHANNEL_MANUFACTURER_MAINTENANCE_DEMANDED	Maintenance Demanded information associated with Channel Diagnosis, Maintenance Demanded information associated with ExtChannel Diagnosis, Maintenance Demanded information associated with Manufacturer Specific Diagnosis

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the Diagnosis Data object that has to be read. The allowed length is from 10 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

List of Diagnosis Data
This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis
This parameter is composed of the following elements.

Block Version Low = 0
This parameter contains the structure identification of the following ASE objects. Block Version Low =0 shall be used, if the IO device only support API 0.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Block Version Low = 1
This parameter contains the structure identification of the following ASE objects. Block Version Low =1 shall be used, if the IO device supports more than API 0.

API
This parameter contains the value of the corresponding attribute of the ASE object.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Diagnosis
This parameter is composed of the following elements.

Block Version Low = 0
This parameter contains the structure identification of the following ASE objects. BlockVersion 1.0 shall be used, if the IO device only supports API 0.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Ext Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value
This parameter contains the value of the corresponding attribute of the ASE object.

Block Version Low = 1
This parameter contains the structure identification of the following ASE objects. BlockVersion 1.1 shall be used, if the IO device supports more than API 0.

API
This parameter contains the value of the corresponding attribute of the ASE object.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Ext Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value
This parameter contains the value of the corresponding attribute of the ASE object.

Qualified Ext Channel Diagnosis
This parameter is composed of the following elements.

Block Version Low = 0
This parameter contains the structure identification of the following ASE objects. BlockVersion 1.0 shall be used, if the IO device only support API 0.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Qualified Ext Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value
This parameter contains the value of the corresponding attribute of the ASE object.

Qualified Channel Qualifier
This parameter contains the value of the corresponding attribute of the ASE object.

Block Version Low = 1
This parameter contains the structure identification of the following ASE objects. BlockVersion 1.1 shall be used, if the IO device supports more than API 0.

API
This parameter contains the value of the corresponding attribute of the ASE object.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Qualified Ext Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value
This parameter contains the value of the corresponding attribute of the ASE object.

Qualified Channel Qualifier
This parameter contains the value of the corresponding attribute of the ASE object.

Manufacturer Specific Diagnosis
This parameter is composed of the following elements.

Block Version Low = 0
This parameter contains the structure identification of the following ASE objects. BlockVersion 1.0 can be used, if the IO device only supports API 0.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

User Structure Identifier
This parameter contains the value of the corresponding attribute of the ASE object.

List of Data
This parameter contains a list of manufacturer specific diagnosis data. A list element consists of the following parameter.

Data
This parameter contains the value of the corresponding attribute of the ASE object.

Block Version Low = 1
This parameter contains the structure identification of the following ASE objects. BlockVersion 1.1 shall be used, if the IO device supports more than API 0. It also can be used, if the IO device only supports API 0.

API
This parameter contains the value of the corresponding attribute of the ASE object.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

User Structure Identifier
This parameter contains the value of the corresponding attribute of the ASE object.

List of Data
This parameter contains a list of manufacturer specific diagnosis data. A list element consists of the following parameter.

Data
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.4.3.3 Diagnosis Event

This service is used to indicate a diagnosis event detected by protocol machines. Table 273 shows the parameters of the service.

Furthermore, the attribute values of the Diagnosis ASE shall be set according to the service parameter. Additionally, an Alarm Notification service shall be issued if it is required (e.g. for Alarm type “Multicast Provider Communication Stopped”).

Table 273 – Diagnosis Event

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
CREP	M	M(=)	
Alarm Item	M	M(=)	
User Structure Identifier	M	M(=)	
List of Channel Diagnosis Data	S	S(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error type	M	M(=)	
List of Diagnosis Data	S	S(=)	
Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Channel Diagnosis Data	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error type	M	M(=)	
Ext Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Ext Channel Diagnosis Data	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error type	M	M(=)	
Ext Channel Error type	M	M(=)	
Ext Channel Add Value	M	M(=)	
Manufacturer Specific Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
User Structure Identifier	M	M(=)	
List of Data	M	M(=)	
Data	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument
The argument shall convey the service specific parameters of the service indication.

AREP
This parameter is the local identifier for the desired AR.

CREP
This parameter is the local identifier for the desired CR.

Alarm Item
This parameter contains alarm specific data if present. The length shall 1 408 octets not exceed.

User Structure Identifier
This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnosis Data
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

List of Diagnosis Data
This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis
This parameter is composed of the following elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter:

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Diagnosis
This parameter is composed of the following elements.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Ext Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value
This parameter contains the value of the corresponding attribute of the ASE object.

Manufacturer Specific Diagnosis
This parameter is composed of the following elements.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

User Structure Identifier
This parameter contains the value of the corresponding attribute of the ASE object.

List of Data
This parameter contains a list of manufacturer specific diagnosis data. A list element consists of the following parameter.

Data
This parameter contains the value of the corresponding attribute of the ASE object.

8.3.4.3.4 Behavior of Diagnosis objects

8.3.4.3.5 General behavior of the Diagnosis object

An entry of a Diagnosis object shall maintain different states for diagnosis, maintenance required, maintenance demanded, and qualified 2 to 32.

8.3.4.3.6 Behavior Diagnosis entry

Table 274 defines the state table for a diagnosis entry.

Table 274 – State table Diagnosis entry

#	Current State	Event /Condition =>Action	Next State
1	NONEXISTENT	**ApplicationDiagnosisDetected (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data)** /ChannelProperties.Maintenance Required=FALSE && ChannelProperties.Maintenance Demanded=FALSE && DiagnosisType= (CHANNELDIAGNOSIS || EXT_CHANNELDIAGNOSI || QUALIFIED_EXT_CHANNELDIAGNOSIS || MANUFACTURER_SPECIFICDIAGNOSIS) => CreateDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelPoperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) UpdateAPDUStatus(For all related ARs) Alarm Item.Diagnosis = BuildDiagnosisAlarmAppears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) A Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	DIAG
2	DIAG	**ApplicationDiagnosisRemoved (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType)** /ChannelProperties.Maintenance Required=FALSE && ChannelProperties.Maintenance Demanded=FALSE && DiagnosisType=(CHANNELDIAGNOSIS || EXT_CHANNELDIAGNOSI || QUALIFIED_EXT_CHANNELDIAGNOSIS)|| MANUFACTURER_SPECIFICDIAGNOSIS) => RemoveChannelDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType) UpdateAPDUStatus(For all related ARs) Alarm Item.Diagnosis = BuildDiagnosisAlarmDisappears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	NONEXISTENT

Table 275 defines the functions used in the state table.

Table 275 – Functions used in state tables

Name	Function
ApplicationDiagnosisDetected (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data)	This event is called by the application process of an IO device to indicate a diagnosis
ApplicationDiagnosisRemoved (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType)	This event is called by the application process of an IO device to indicate that a diagnosis is not longer existing
CreateDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelPoperties)	invokes a diagnosis entry of the given type within the diagnosis object
RemoveDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelPoperties)	Deletes a diagnosis entry of the given type within the diagnosis object
UpdateAPDUStatus(For all related ARs)	Sets the Problem Indicator Flag to TRUE for all related ARs if at least one diagnosis appear entry for the API/module/submodule exist, otherwise it shall be set to FALSE
BuildDiagnosisAlarmAppears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data)	Returns the diagnosis entry of the given format for the alarm notification (Channel Properties. Specifier := APPEAR)
BuildDiagnosisAlarmDisappears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ExtChannelErrorType)	Returns the diagnosis entry of the given format for the alarm notification, the diagnosis entry (Channel Properties. Specifier) shall contain the summarized diagnosis info for the whole API/module/submodule/channel to select DISAPPEAR or DISAPPEAR_OTHER_EXIST
BuildSummarizedAlarmSpecifier(API, Slot, Subslot)	Returns the alarm specifier entry for the alarm notification, the alarm specifier shall contain the summarized alarm info for the whole API/module/submodule (Submodule Diagnosis State, Submodule Channel Diagnosis, Submodule Manufacturer Specific Diagnosis) and for all API/modules/submodules of the related AR (AR Diagnosis State)
BuildSummarizedMaintenanceStatus(API, Slot, Subslot)	Returns the Maintenance Status entry for the alarm notification, it shall contain the summarized maintenance information of the API/module/submodule (required, demanded, or qualified 2 to 31) if no maintenance is present nothing shall be returned and the optional service parameter shall be omitted in the alarm notification
GetAREPAlarmPriority(AREP)	Returns the alarm priority for the alarm type dedicated to a certain AR

8.3.4.3.7 Behavior maintenance required entry

Table 276 defines the state table for a maintenance required entry.

Table 276 – State table maintenance required entry

#	Current State	Event /Condition =>Action	Next State
1	NONEXISTENT	**ApplicationDiagnosisDetected (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data)** /ChannelProperties.Maintenance Required=TRUE && ChannelProperties.Maintenance Demanded=FALSE && DiagnosisType= (CHANNELDIAGNOSIS || EXT_CHANNELDIAGNOSI || QUALIFIED_EXT_CHANNELDIAGNOSIS || MANUFACTURER_SPECIFICDIAGNOSIS) => CreateDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelPoperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Item.Diagnosis = BuildDiagnosisAlarmAppears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	MAINTENANCE_REQ
2	MAINTENANCE_REQ	**ApplicationDiagnosisRemoved (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType)** /ChannelProperties.Maintenance Required=TRUE && ChannelProperties.Maintenance Demanded=FALSE && DiagnosisType=(CHANNELDIAGNOSIS || EXT_CHANNELDIAGNOSI || QUALIFIED_EXT_CHANNELDIAGNOSIS || MANUFACTURER_SPECIFICDIAGNOSIS) => RemoveChannelDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType) Alarm Item.Diagnosis = BuildDiagnosisAlarmDisappears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	NONEXISTENT

Table 275 defines the functions used in the state table.

8.3.4.3.8 Behavior maintenance demanded entry

Table 277 defines the state table for a maintenance demanded entry.

Table 277 – State table maintenance demanded entry

#	Current State	Event /Condition =>Action	Next State
1	NONEXISTENT	**ApplicationDiagnosisDetected (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data)** /ChannelProperties.Maintenance Required=FALSE && ChannelProperties.Maintenance Demanded=TRUE && DiagnosisType= (CHANNELDIAGNOSIS || EXT_CHANNELDIAGNOSI || QUALIFIED_EXT_CHANNELDIAGNOSIS || MANUFACTURER_SPECIFICDIAGNOSIS) => CreateDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelPoperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Item.Diagnosis = BuildDiagnosisAlarmAppears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) A Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	MAINTENANCE_DEM
2	MAINTENANCE_DEM	**ApplicationDiagnosisRemoved (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType)** /ChannelProperties.Maintenance Required=FALSE && ChannelProperties.Maintenance Demanded=TRUE && DiagnosisType=(CHANNELDIAGNOSIS || EXT_CHANNELDIAGNOSI || QUALIFIED_EXT_CHANNELDIAGNOSIS || MANUFACTURER_SPECIFICDIAGNOSIS) => RemoveChannelDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType) Alarm Item.Diagnosis = BuildDiagnosisAlarmDisappears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	NONEXISTENT

Table 275 defines the functions used in the state table.

8.3.4.3.9 Behavior qualified entry

Table 278 defines the state table which exists for each qualified entry (2 to 31).

Table 278 – State table qualified entry

#	Current State	Event /Condition =>Action	Next State
1	NONEXISTENT	**ApplicationDiagnosisDetected (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data=NIL)** /ChannelProperties.Maintenance Required=TRUE && ChannelProperties.Maintenance Demanded=TRUE && DiagnosisType= (QUALIFIED_EXT_CHANNELDIAGNOSIS) => CreateDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ChannelPoperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Item.Diagnosis = BuildDiagnosisAlarmAppears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data=NIL) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	QUALIFIED
2	QUALIFIED	**ApplicationDiagnosisRemoved (API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType, QualifiedChannelQualifier)** /ChannelProperties.Maintenance Required=FALSE && ChannelProperties.Maintenance Demanded=TRUE && DiagnosisType=(CHANNELDIAGNOSIS || EXT_CHANNELDIAGNOSI || QUALIFIED_EXT_CHANNELDIAGNOSIS || MANUFACTURER_SPECIFICDIAGNOSIS) => RemoveChannelDiagnosisEntry(API, Slot, Subslot,ChannelNumber, DiagnosisType, ChannelErrorType, ExtChannelErrorType) Alarm Item.Diagnosis = BuildDiagnosisAlarmDisappears(API, Slot, Subslot,ChannelNumber, Alarm Item Format, ChannelErrorType, ChannelProperties, ExtChannelErrorType, AddInfo, QualifiedChannelQualifier, Data) Alarm Specifier=BuildSummarizedAlarmSpecifier(API, Slot, Subslot) Alarm Item. Maintenance Status=BuildSummarizedMaintenanceStatus(API, Slot, Subslot) Alarm Priority=GetAREPAlarmPriority(AREP) Alarm Notification.req(AREP, API, Alarm Priority, Alarm type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)	NONEXISTENT

Table 275 defines the functions used in the state table.

8.3.4.3.10 Optimization for Diagnosis objects

An diagnosis object shall be fully addressed with API, Slot Number, Subslot Number, Channel Number, Channel Error type, Ext Channel Error type, and Channel Properties (using only Maintenance required, Maintenance demanded and Specifier).

8.3.4.3.10.1 Subslot Number level

If all diagnosis objects of the address level subslot (one Subslot Number) are deleted at once, then a Diagnosis disappears alarm without an alarm item shall be generated.

This signals, that all subsequent diagnosis, maintenance (required and demanded) and qualified are gone.

8.3.4.3.10.2 Channel Error type level

If all diagnosis objects of the address level Channel Error type (one Channel Error type) are deleted at once, then an alarm item with the following content shall be generated:

- Channel Number
- Channel Properties
  - Type
  - Direction
  - Maintenance Required and Maintenance Demanded shall be zero
  - Specifier shall be zero
- Channel Error type

This signals, that all subsequent diagnosis, maintenance (required and demanded) and qualified are gone.

NOTE Subsequent in this case means, also all subsequent diagnosis objects addressed by Ext Channel Error Types are gone.

8.3.5 Alarm ASE

8.3.5.1 Overview

The alarm model allows the transfer of an alarm from the IO device to the assigned IO controller or vice versa and the explicit acknowledgement of the alarm.

For the transmission of an alarm the following conditions have to be fulfilled:

- the IO AR shall be established
- the source submodule shall be owned by the IO AR
- the source submodule shall be existent (no empty subslot and no wrong submodul)
- a detected alarm shall always be issued, the sink shall response with a negative acknowledgment "not supported" in case no handler for the Alarm type is installed

The following Alarm Types are defined. They may be extended specific to the manufacturer:

Diagnosis Alarm
A diagnosis alarm signals an event within a submodule, for instance over-temperature, short circuit, etc. The content of the alarm is defined by the Diagnosis ASE for this type. It shall also contain maintenance status information and/or maintenance status change information if present.

NOTE 1 The Diagnosis Alarm conveys one or more alarm blocks containing appearing and/or disappearing diagnosis and/or maintenance required and/or demanded.

Process Alarm
A process alarm signals the occurrence of an event in the connected process, for instance upper limit value exceeded.

Pull Alarm / Pull Module Alarm
A slot signals the withdrawal of a submodule/module or change in configuration (reduction).

Plug Alarm
A slot signals the insertion of a submodule, a new need for parametrization, or a change in configuration (addition).

Status Alarm
A status alarm signals a change in the state of a submodule, for instance run, stop or ready.

Update Alarm
An update alarm signals the change of a parameter in a submodule e.g. by a local operation or a remote access.

Redundancy Alarm
A redundancy alarm signals the fault of one IO controller to the remaining IO controller for redundant IO ARs. It shall also contain maintenance status information and/or maintenance status change information if present.

Controlled by supervisor
A slot signals the logical withdrawal of a submodule by the IO supervisor. The actions shall be according to the Pull Alarm.

Released Alarm
A slot signals the logical insertion of a submodule by the IO supervisor. The actions shall be according to the Plug Alarm.

Plug Wrong Submodule Alarm
A slot signals the insertion of a wrong submodule or a change in configuration (addition).

Return of Submodule Alarm
A slot signals that a submodule is ready to switch its IOCS/IOPS from "BAD" to "GOOD" again without new parameterization.

Diagnosis disappears Alarm
A diagnosis disappears alarm signals a disappearing diagnosis event within a submodule. If this alarm is conveyed without an Alarm Item all previous diagnosis is gone. The content of the alarm is defined by the Diagnosis ASE for this type. It shall also contain maintenance status information and/or maintenance status change information if present. If this alarm is conveyed without a Maintenance Item all previous maintenance is gone.

NOTE 2 It is recommended to use the Diagnosis disappears Alarm to signal an error free alarm source, only.

Multicast Communication Mismatch
A multicast consumer submodule signals that communication relationship to the associated multicast provider is failed. It shall also contain maintenance status information and/or maintenance status change information if present.

Port Data Change Notification Alarm
A port submodule signals that port data has been changed. It shall also contain maintenance status information and/or maintenance status change information if present.

Sync Data Change Notification Alarm
An interface submodule signals that synchronization data has been changed. It shall also contain maintenance status information and/or maintenance status change information if present.

Isochronous Mode Problem Notification Alarm
The application signals that problems with isochronously execution have been detected. It shall also contain maintenance status information and/or maintenance status change information if present.

Time Data Change Notification Alarm
An interface submodule signals that the time synchronization data has been changed. It shall also contain maintenance status information and/or maintenance status change information if present.

Upload and Storage Notification Alarm
A submodule signals that Record Data objects has been changed and shall be uploaded to the IO controller.

Furthermore, manufacturer specific alarms may be used and alarms are reserved for profile specific definitions.

8.3.5.2 Alarm class specification

8.3.5.2.1 Template

An Alarm object is described by the following template:

ASE: **Alarm ASE**
CLASS: **Alarm**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Identifier
1.1	(m)	Attribute:	Alarm type
1.2	(m)	Attribute:	Slot Number
1.3	(m)	Attribute:	Subslot Number
2	(m)	Attribute:	Alarm Specifier
2.1	(m)	Attribute:	Sequence Number
2.2	(m)	Attribute:	Channel Diagnosis
2.3	(m)	Attribute:	Manufacturer Specific Diagnosis
2.4	(m)	Attribute:	Submodule Diagnosis State
2.5	(m)	Attribute:	AR Diagnosis State
3	(m)	Attribute:	Module Ident Number
4	(m)	Attribute:	Submodule Ident Number
5	(o)	Attribute:	Alarm Item
5.1	(m)	Attribute:	User Structure Identifier
5.2	(s)	Attribute:	List of Data
5.2.1	(m)	Attribute:	Data
5.3	(s)	Attribute:	List of Channel Diagnosis Data
5.3.1	(m)	Attribute:	Channel Number
5.3.2	(m)	Attribute:	Channel Properties
5.3.3	(m)	Attribute:	Channel Error type
5.4	(s)	Attribute:	List of Diagnosis Data
5.4.1	(s)	Attribute:	Channel Diagnosis
5.4.1.1	(m)	Attribute:	Slot Number
5.4.1.2	(m)	Attribute:	Subslot Number
5.4.1.3	(m)	Attribute:	List of Channel Diagnosis Data
5.4.1.3.1	(m)	Attribute:	Channel Number
5.4.1.3.2	(m)	Attribute:	Channel Properties
5.4.1.3.3	(m)	Attribute:	Channel Error type
5.4.2	(s)	Attribute:	Ext Channel Diagnosis
5.4.2.1	(m)	Attribute:	Slot Number
5.4.2.2	(m)	Attribute:	Subslot Number
5.4.2.3	(m)	Attribute:	List of Ext Channel Diagnosis Data
5.4.2.3.1	(m)	Attribute:	Channel Number
5.4.2.3.2	(m)	Attribute:	Channel Properties
5.4.2.3.3	(m)	Attribute:	Channel Error type
5.4.2.3.4	(m)	Attribute:	Ext Channel Error type
5.4.2.3.5	(m)	Attribute:	Ext Channel Add Value
5.4.3	(s)	Attribute:	Manufacturer Specific Diagnosis
5.4.3.1	(m)	Attribute:	Slot Number
5.4.3.2	(m)	Attribute:	Subslot Number
5.4.3.3	(m)	Attribute:	Channel Number
5.4.3.4	(m)	Attribute:	Channel Properties
5.4.3.5	(m)	Attribute:	User Structure Identifier
5.4.3.6	(m)	Attribute:	List of Data
5.4.3.6.1	(m)	Attribute:	Data
6	(m)	Attribute:	Related AREP
7	(m)	Attribute:	Alarm Priority
8	(o)	Attribute:	Maintenance Status

8.1	(m)	Attribute:	Maintenance Required
8.2	(m)	Attribute:	Maintenance Demanded
8.3	(m)	Attribute:	Maintenance Qualifier 2
8.4	(m)	Attribute:	Maintenance Qualifier 3
8.5	(m)	Attribute:	Maintenance Qualifier 4
8.6	(m)	Attribute:	Maintenance Qualifier 5
8.7	(m)	Attribute:	Maintenance Qualifier 6
8.8	(m)	Attribute:	Maintenance Qualifier 7
8.9	(m)	Attribute:	Maintenance Qualifier 8
8.10	(m)	Attribute:	Maintenance Qualifier 9
8.11	(m)	Attribute:	Maintenance Qualifier 10
8.12	(m)	Attribute:	Maintenance Qualifier 11
8.13	(m)	Attribute:	Maintenance Qualifier 12
8.14	(m)	Attribute:	Maintenance Qualifier 13
8.15	(m)	Attribute:	Maintenance Qualifier 14
...			
8.31	(m)	Attribute:	Maintenance Qualifier 31
9	(o)	Attribute:	Upload and Storage
9.1.1	(m)	Attribute:	List of Records
9.1.1.1	(m)	Attribute:	Record Number
9.1.1.2	(m)	Attribute:	Record Length

SERVICES:

1	(m)	OpsService:	Alarm Notification
2	(m)	OpsService:	Alarm Ack

8.3.5.2.2 Attributes

Identifier

This key attribute is composed of Slot Number, Subslot Number, and Alarm type to define to which structural slot (module), subslot and to which Alarm type the Alarm object belongs. This Identifier is unique for each application process within the IO device and shall not be used by another object.

Alarm type

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

The allowed values of Alarm type are shown in Table 279.

Table 279 – Alarm type

Alarm type	Usage	
Diagnosis	The value Diagnosis shall be used to indicate appearing or disappearing items for one or more channels	
Process	Indicates an alarm from the controlled process (usage application and process specific)	
Pull	Shall indicate an alarm if a used module/submodule is pulled	Mandatory
Pull module	Shall indicate an alarm if a used module is pulled	Optional
Plug	Shall indicate an alarm if a requested submodule is plugged	Mandatory
Status	Indicates a status change within a submodule	
Update	Indicates a change of parameter for a submodule	
Redundancy	Indicates a second IO controller that the primary one has failed	Mandatory for IO controller redundancy
Controlled	Shall indicate an alarm if a used submodule has been locked by an IO supervisor	Mandatory
Released	Shall indicate an alarm if a requested submodule has been unlocked by an IO supervisor, IO Controller or by IO Device local means	Mandatory
Plug Wrong Submodule	Shall indicate an alarm if a wrong submodule for a requested module/submodule is plugged	Mandatory
Diagnosis Disappears	The value Diagnosis Disappears shall only be used - to indicate all disappearing without an Alarm Item (optimisation) - to indicate detailed disappearing items	
Return Of Submodule	Shall indicate a change of IOCS/IOPS from BAD to GOOD for a submodule	Mandatory
Profile Specific	Shall be used according upcoming PNO Profile Guidelines if an IO devices complies to a specific device profile	
Multicast Communication Mismatch Notification	A multicast consumer shall indicate that the multicast CR has had a problem	
Port Data Change Notification	Indicates a change of port data e.g. link up or down	Mandatory
Sync Data Changed Notification	Indicates a change of clock synchronization	
Isochronous Mode Problem Notification	Indicates a problem of isochronous applications, e.g. started to late	
Network Component Problem Notification	Indicates a problem of network components, e.g. frame dropped no resources	Mandatory
Time Data Changed Notification	Indicates a change of time synchronization	
Upload and Retrieval Notification	Indicates a request for upload and storage of Record Data objects	Optional
Manufacturer Specific	Values from 0x0020 to 0x007F may be used for manufacturer specific alarm types	
NOTE The Diagnosis Disappears is only recommended to convey disappearing of manufacturer specific status items. It may also be used as optimization to convey the information that all previosly items are gone.		

Slot Number
This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. It shall contain the physical or logical slot which issues the alarm.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number
This attribute defines the subslot number as a local identifier. This number is unique within a device. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. It shall contain the physical or logical subslot which issues the alarm.

Attribute type: Unsigned16

Allowed values: 0 to 0x8FFF are allowed

NOTE 1 The value 0 is only used in conjunction with pull alarm.

Alarm Specifier
This attribute consists of the following attributes.

Sequence Number
This attribute defines the sequence number of the alarm. It shall be incremented with every Alarm Notification service. By means of the Sequence Number the receiver detects duplications and reflects the value of the field in the Alarm Ack.

Allowed values: 0 to 2 047

Channel Diagnosis
This attribute shall only be used for the Alarm type values Diagnosis, Redundancy, Port data change notification, Sync data change notification, Isochronous mode problem notification, and Multicast communication mismatch. For all other Alarm Types this attribute is not valid.

Table 280 shows the allowed values.

Table 280 – Channel Diagnosis

Value	Meaning
NO_CHANNEL_DIAG	Means that the Subslot Number does not include a Channel Diagnosis
CHANNEL_DIAG	Means that the Subslot Number includes at least one Channel Diagnosis

Manufacturer Specific Diagnosis
This attribute shall only be used for the Alarm type values Diagnosis, Redundancy, Port data change notification, Sync data change notification, Isochronous mode problem notification, and Multicast communication mismatch. For all other Alarm Types this attribute is not valid.

Table 281 shows the allowed values.

Table 281 – Manufacturer Specific Diagnosis

Value	Meaning
NO_MAN_DIAG	Means that the Subslot Number does not include a Manufacturer Specific Diagnosis
MAN_DIAG	Means that the Subslot Number includes at least one Manufacturer Specific Diagnosis

Submodule Diagnosis State
This attribute shall only be used for the Alarm type values Diagnosis, Redundancy, Port data change notification, Sync data change notification, Isochronous mode problem notification, and Multicast communication mismatch. For all other Alarm Types this attribute is not valid.

Table 282 shows the allowed values.

Table 282 – Submodule Diagnosis State

Value	Meaning
NO_DIAG	Error free Furthermore, it indicates that all reported diagnosis have been cleared. An individual “disappears” notification can be omitted. However, even in this case a channel diagnosis locally treated as status and/or a Manufacturer Specific Diagnosis treated as status may be present.
DIAG	At least one diagnosis exists at the submodule. It could be one or more channel diagnosis and/or one or more Manufacturer Specific Diagnosis.

AR Diagnosis State
This attribute shall only be used for the Alarm type values Diagnosis, Redundancy, Port data change notification, Sync data change notification, Isochronous mode problem notification, and Multicast communication mismatch. For all other Alarm Types this attribute is not valid. The AR Diagnosis State shall be correspond to the current state of

the IO Data ASE attribute Problem Indicator Flag, which is related to all modules/submodules that belong to the AR.

Table 283 shows the allowed values.

Table 283 – AR Diagnosis State

Value	Meaning
NO_DIAG	Error free Furthermore, it indicates that all reported diagnosis related to the AR have been cleared. An individual "disappears" notification can be omitted. However, even in this case a channel diagnosis locally treated as status and/or a Manufacturer Specific Diagnosis treated as status may be present.
DIAG	At least one diagnosis exists at one submodule related to the AR. It could be one or more channel diagnosis and/or one or more Manufacturer Specific Diagnosis.

Module Ident Number
This attribute contains the module identification.

Attribute type: Unsigned32

Allowed Values: 1 to 0xFFFFFFFF

NOTE 2 The attribute value is manufacturer specific.

Submodule Ident Number
This attribute contains the submodule identification.

Attribute type: Unsigned32

NOTE 3 The attribute value is manufacturer specific.

Alarm Item
This attribute is composed of the following attributes.

User Structure Identifier
This attribute identifies the selection of the Alarm Notification data. The allowed values are shown in Table 284.

Table 284 – User Structure Identifier

Value	Meaning / Usage
Manufacturer Specific	In conjunction with alarm type Diagnosis Manufacturer Specific Diagnosis in Alarm Notification and Diagnosis Data. In conjunction with other alarm types the usage is manufacturer specific.
Profile Specific	The meaning is defined in profile specifications.
Channel Diagnosis	Shall only be used in conjunction with Channel Diagnosis in Alarm Notification and Diagnosis Data.
Extended Channel Diagnosis	Shall only be used in conjunction with Extended Channel Diagnosis in Alarm Notification and Diagnosis Data.
Qualified Extended Channel Diagnosis	Shall only be used in conjunction with Qualified Extended Channel Diagnosis in Alarm Notification and Diagnosis Data.
Multiple	Shall only be used in conjunction with alarm type, which comply the Block Structure. It shall also be used with Multicast Consumer Info Data, List of Isochronous Mode Info Data, List of Port Info Data, List of Sync Info Data
Maintenance	Maintenance shall be used in conjunction with following AlarmTypes: Diagnosis, Redundancy, DiagnosisDisappears, multicast communication mismatch, Port data change notification, Sync data change notification, isochronous mode problem notification, Network component problem notification, Time data changed notification Furthermore, the AlarmNotification shall only convey this block if the alarm source includes at least one Maintenance Required entry, or one Maintenance Demanded entry, or one Qualifier_x entry.
Upload and Retrival	Upload&Retrival shall be used in conjunction with Upload and storage notification.

Manufacturer specific structures shall only be used if Channel Diagnosis, or Extended Channel Diagnosis or Qualified Extended Channel Diagnosis does not cover the diagnosis information.

List of Data
This attribute contains a list of user specific alarm data.

The length of the user data shall 1 408 octets not exceed.

A list element is composed of the following attributes:

Data
This attribute contains user specific alarm data.

Attribute type: One of the following data types shall be used:
Boolean, Binary Date, Integer, Time of Day, Unsigned, Time-Difference, Floating Point, Network Time, VisibleString, Network Time Difference, OctetString

List of Channel Diagnosis Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF user specific channel number, 0x8000 submodule

Channel Properties
This attribute has the same definition as specified in the Diagnosis ASE with an exception for the attribute Specifier.

Specifier
This attribute indicates that Channel Diagnosis appears or disappears, that Maintenance Required appears or disappears, or that Maintenance Demanded appears or disappears. The semantic depends on the values of the attributes Maintenance Required and Maintenance Demanded. The dependencies are shown in Table 285.

Allowed values: APPEARS, DISAPPEARS, DISAPPEARS_BUT_OTHER_REMAIN, ALL_DISAPPERS

Table 285 – Semantics of Specifier

Maintenance Required	Maintenance Demanded	Specifier	Meaning
NO_MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	APPEARS	Diagnosis appears
NO_MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	ALL_DISAPPEARS	In conjunction with Channel diagnosis, all subsequent[a] Diagnosis, Maintenance (required and demanded) and Qualified disappears
NO_MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	DISAPPEARS	Diagnosis disappears
NO_MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	DISAPPEARS_BUT_OTHER_REMAIN	Diagnosis disappears but the channel still has other diagnosis
MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	APPEARS	Maintenance Required information appears
MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	DISAPPEARS	Maintenance Required information disappears
MAINTENANCE_REQUIRED	NO_MAINTENANCE_DEMANDED	DISAPPEARS_BUT_OTHER_REMAIN	Maintenance Required information disappears but the channel still has other Maintenance Required
NO_MAINTENANCE_REQUIRED	MAINTENANCE_DEMANDED	APPEARS	Maintenance Demanded information appears
NO_MAINTENANCE_REQUIRED	MAINTENANCE_DEMANDED	DISAPPEARS	Maintenance Demanded information disappears
NO_MAINTENANCE_REQUIRED	MAINTENANCE_DEMANDED	DISAPPEARS_BUT_OTHER_REMAIN	Maintenance Demanded information disappears but the channel still has other Maintenance Demanded

[a] Subsequent means, that for this particular ChannelErrorType all ExtChannelErrorTypes (and the ChannelErrorType itself) with diagnosis, maintenance required, maintenance demanded, and qualified changes to OK.

Channel Error type
This attribute indicates the error type of the Channel Related Diagnosis.

Attribute type: Unsigned16

Allowed values: same values as specified in the Diagnosis ASE.

List of Diagnosis Data
This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis
This parameter contains channel related diagnosis data and consists of the following elements.

Slot Number
This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order

but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number
This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE The Subslot Number zero does not contain any diagnosis.

List of Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF user specific channel number, 0x8000 submodule

Channel Properties
This attribute has the same definition as specified in the Diagnosis ASE.

Channel Error type
This attribute indicates the error type of the Channel Related Diagnosis.

Attribute type: Unsigned16

Allowed values: same values as specified in the Diagnosis ASE.

Ext Channel Diagnosis
This attribute contains an extended channel diagnosis. It consists of the following attributes.

Slot Number
This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number
This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE The Subslot Number zero does not contain any diagnosis.

List of Ext Channel Diagnostic Data
This parameter contains a list of extended channel diagnosis data. A list element consists of the following parameter.

Channel Number
This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF user specific channel number, 0x8000 submodule

Channel Properties
This attribute has the same definition as specified in the Diagnosis ASE.

Channel Error type
This attribute indicates the error type of the Channel Related Diagnosis.

Attribute type: Unsigned16

Allowed values: same values as specified in the Diagnosis ASE.

Ext Channel Error type
This attribute depends on the attribute Channel Error type and indicates an extended error type. The allowed values are shown in Table 268.

Attribute type: Unsigned16

Ext Channel Add Value
This attribute contains an additional value.

Attribute type: Unsigned32

Manufacturer Specific Diagnosis
This attribute contains a manufacturer specific diagnosis and consists of the following attributes.

Slot Number
This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number
This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE 4 The Subslot Number zero does not contain any diagnosis.

Channel Number
This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF user specific channel number, 0x8000 submodule

Channel Properties
This attribute has the same definition as specified in the Diagnosis ASE.

User Structure Identifier
This attribute indicates the structure of the Manufacturer Specific Diagnosis Data attribute.

Attribute type: Unsigned16

Allowed values: MANUFACTURER_SPECIFIC_1 up to MANUFACTURER_SPECIFIC_0x7FFF, CHANNEL_DIAGNOSIS, EXT_CHANNEL_DIAGNOSIS, QUALIFIED_EXT_CHANNEL_DIAGNOSIS, MULTIPLE_DIAGNOSIS

List of Data
This attribute contains a list of manufacturer specific diagnosis data. A list element consists of the following attributes.

Data
This attribute contains manufacturer specific diagnosis data.

Attribute type: One of the following data types shall be used: Boolean, Binary Date, Integer, Time of Day, Unsigned, Time-Difference, Floating Point, Network Time, VisibleString, Network Time Difference, OctetString

Related AREP
This attribute contains the AREP of the related AR for transmission.

Attribute type: Unsigned32

Alarm Priority
This attribute contains the priority of the alarm.

Allowed Values: ALARM_HIGH, ALARM_LOW

Maintenance Status
This optional attribute contains a maintenance status and consists of the following attributes. It shall be only conveyed if at least one of the attributes below has maintenance information. Otherwise it shall be omitted.

Maintenance Required
This attribute contains the value MAINTENANCE_REQUIRED if at least one channel of the alarm source (API, Slot, Subslot) contains maintenance required information.

Allowed Values: MAINTENANCE_REQUIRED, NO_MAINTENANCE_REQUIRED

Maintenance Demanded
This attribute contains the value MAINTENANCE_DEMANDED if at least one channel of the alarm source (API, Slot, Subslot) contains maintenance demanded information.

Allowed Values: MAINTENANCE_DEMANDED, NO_MAINTENANCE_DEMANDED

Qualifier 2
This attribute contains the value QUALIFIER_2_SET if at least one channel of the alarm source (API, Slot, Subslot) contains qualified maintenance information. The semantic shall be defined by profiles.

Allowed Values: QUALIFIER_2_SET, QUALIFIER_2_NOT_SET

...

Qualifier 31
This attribute contains the value QUALIFIER_31_SET if at least one channel of the alarm source (API, Slot, Subslot) contains qualified maintenance information. The semantic shall be defined by profiles.

Allowed Values: QUALIFIER_31_SET, QUALIFIER_31_NOT_SET

Upload and Retrieval
With this optional attribute an IO device indicates the existence of Record Data objects which shall be stored at the IO controller. The IO controller shall upload and permanent store the Record Data objects. At every following connect sequence and for every following plug and released sequence the IO controller shall convey this additional Record Data objects to the source of the stored Record Data objects. Record Data objects which are part of the GSDML

description and conveyed from the engineering tools to the IO controller shall not be part of the List of Records. It consists of the following attributes.

List of Records
This attribute contains a list of Record Data objects. A list element consists of the following attributes.

Record Number
This attribute contains the parameter Index of a Record Data object as shown in the Record Data ASE.

Allowed Values: 0 – 0xFFFF

Record Length
This attribute contains the parameter Length of a Record Data object as shown in the Record Data ASE.

Allowed Values: 2^0 to 2^{32}-256

8.3.5.2.3 Invocation of the Alarm object

Several Alarm objects can be invoked in an IO device or an IO controller.

8.3.5.3 Alarm service specification

8.3.5.3.1 Alarm Notification

This service is used to transfer an alarm notification from the IO device to the IO controller and optional vice versa. This service shall only be used in conjunction with the IO AR. Table 286 shows the parameters of the service.

Table 286 – Alarm Notification

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
API	M	M(=)	
Alarm Priority	M	M(=)	
Alarm type	M	M(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Alarm Specifier	M	M(=)	
Module Ident Number	M	M(=)	
Submodule Ident Number	M	M(=)	
Alarm Item	U	U(=)	
User Structure Identifier	M	M(=)	
List of Data	S	S(=)	
Data	M	M(=)	
List of Channel Diagnosis Data	S	S(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error type	M	M(=)	
List of Diagnosis Data	S	S(=)	
Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Channel Diagnosis Data	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error type	M	M(=)	
Ext Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Ext Channel Diagnosis Data	M	M(=)	

Parameter name	Req	Ind	Cnf
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error type	M	M(=)	
Ext Channel Error type	M	M(=)	
Ext Channel Add Value	M	M(=)	
Manufacturer Specific Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
User Structure Identifier	M	M(=)	
List of Data	M	M(=)	
Data	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter is the identifier for the desired application process.

Alarm Priority
This parameter contains the value ALARM_HIGH or ALARM_LOW.

Alarm type
This parameter contains the value of the attribute Alarm type of the Alarm Data object.

Slot Number
This parameter contains the value of the attribute Slot Number of the Alarm Data object.

Subslot Number
This parameter contains the value of the attribute Subslot Number of the Alarm Data object.

Alarm Specifier
This parameter contains the value of the attribute Alarm Specifier of the Alarm Data object.

Module Ident Number
This parameter contains the value of the attribute Module Ident Number of the Alarm Data object.

Submodule Ident Number
This parameter contains the value of the attribute Submodule Ident Number of the Alarm Data object.

Alarm Item
This parameter contains alarm specific data if present. The length shall 1 408 octets not exceed.

User Structure Identifier
This parameter contains the value of the corresponding attribute of the ASE object.

List of Data
This parameter contains the value of the corresponding attribute of the ASE object.

Data
This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnosis Data
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

List of Diagnosis Data
This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis
This parameter is composed of the following elements.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Diagnosis
This parameter is composed of the following elements.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Ext Channel Diagnostic Data
This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error type
This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value
This parameter contains the value of the corresponding attribute of the ASE object.

Manufacturer Specific Diagnosis
This parameter is composed of the following elements.

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number
This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties
This parameter contains the value of the corresponding attribute of the ASE object.

User Structure Identifier
This parameter contains the value of the corresponding attribute of the ASE object.

List of Data
This parameter contains a list of manufacturer specific diagnosis data. A list element consists of the following parameter.

Data
This parameter contains the value of the corresponding attribute of the ASE object.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Status
This parameter contains the reason for the failure.

Allowed values: AR_NOT_ESTABLISHED, ALARM_TYPE_NOT_SUPPORTED, LIMIT_EXPIRED, SEQUENCE_NR_PENDING

8.3.5.3.2 Alarm Ack

This service is used to acknowledge the reception of an alarm notification which has been previously received. This service shall only be used in conjunction with the IO or Supervisor AR.

Table 287 shows the parameters of the service.

Table 287 – Alarm Ack

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
API	M	M(=)	
Alarm type	M	M(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Alarm Specifier	M	M(=)	
PNIO Status	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter is the identifier for the desired application process.

Alarm type
This parameter contains the value of the attribute Alarm type of the previously received Alarm Data object.

Slot Number
This parameter contains the value of the attribute Slot Number of the of the previously received Alarm Data object.

Subslot Number
This parameter contains the value of the attribute Subslot Number of the of the previously received Alarm Data object.

Alarm Specifier
This parameter contains the value of the attribute Alarm Specifier of the of the previously received Alarm Data object.

PNIO Status
This parameter contains the error status.

Allowed values: NO_ERROR, ALARM_TYPE_NOT_SUPPORTED, WRONG_SUBMODULE_STATE

Attribute type: Unsigned32

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

8.3.5.4 Behavior of Alarm objects

8.3.5.4.1 General behavior of the alarm sink

The application behavior based on flow control of alarm transmission, IOPS, IOCS and Problem Indicator of the IO Data ASE. These mechanisms make it convenient for the application to process the alarm.

NOTE The application process may not handle the alarm immediately by interrupting other processes. It may also poll for queued alarms to handle these in an appropriate time window.

Each application process shall maintain a receiving queue for each Alarm type. If used the queue size is determined by the flow control mechanisms. With the processing of the alarm the acknowledgment shall be sent to the source to enable the sending of further alarms of this type.

Alarm queues shall be cleared in case of AR termination and pull/plug alarms.

8.3.5.4.2 General behavior of the alarm source

Figure 90 shows the model at the alarm source. The source is divided into the IO device (IOD), the application process and the alarm source.

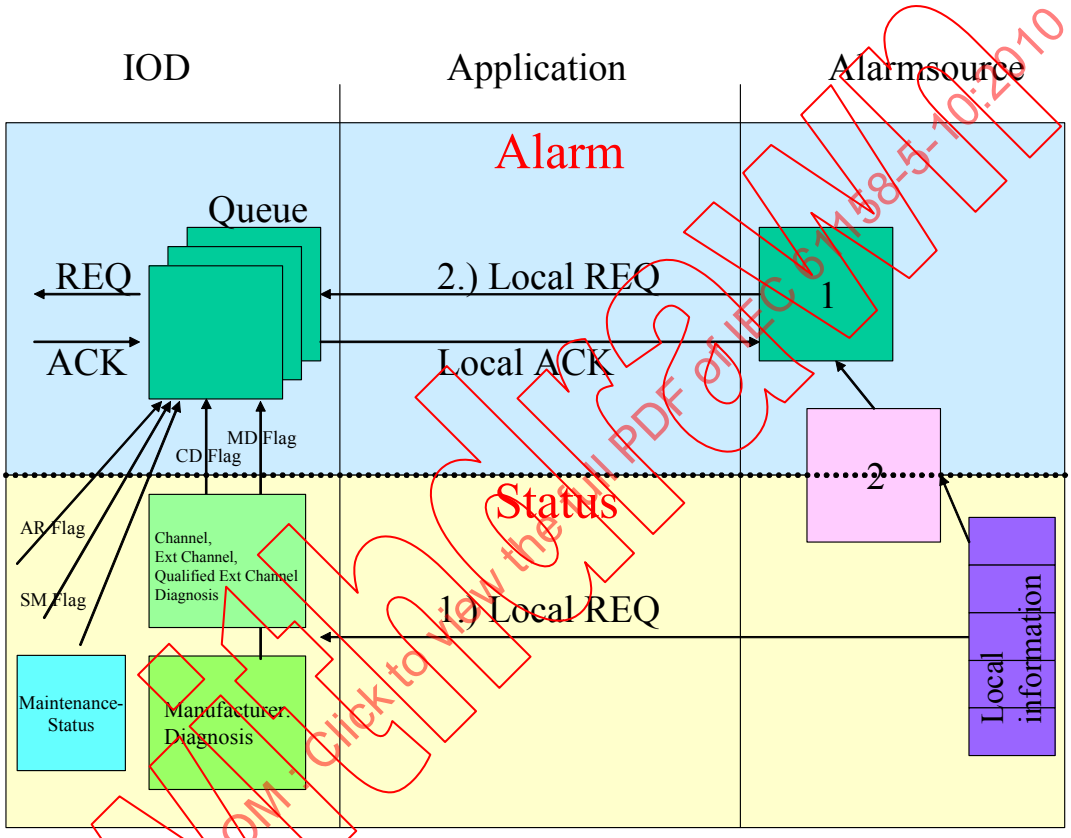


Figure 90 – Example of a resource model at the alarm source

The model shows how to handle event overflow at the source. Event overflow may cause by flow control because the sink determines how fast alarms are handled. A mixture of queue and buffer model solves that problem for the Alarm ASE.

The alarm source maintains local information of the state. If, for example, the status changes from GOOD to BAD it is notified. It maintains at least two buffers for alarms. The first buffer is used to transport the state transition. The second buffer contains the current state. A new state transition will not be reported as long as the acknowledgement is not received in case of only two buffers. But the current state may be read with diagnosis information at any time.

The IOD maintains two queues for each priority. A new alarm of the same priority is blocked as long as the acknowledgement is outstanding.

Furthermore, alarm queues shall be cleared in case of AR termination and pull/plug alarms.

8.3.5.4.3 Behavior for diagnosis changes

Diagnosis alarms are the means to report and acknowledge diagnosis changes. It is strongly recommended to use Channel Diagnoses or Extended Channel Diagnosis because of the well-defined structure.

The recommended sequence for signaling a diagnosis is:

First, update the diagnosis ASE

Second, issue an alarm

Diagnosis alarms include information of the alarm source and the direction of changes. The attribute Alarm Specifier contains information about the direction e.g. appears or disappears.

All channel diagnosis are gone is reported by an alarm with the parameter Alarm Specifier Submodule Diagnosis State “NO_ERROR”. It may save a lot of individual messages.

8.3.6 Context ASE

8.3.6.1 Overview

The Context ASE provides a set of services to establish application relationships, parameterize and configure the AP of an IO device itself and its slots/subslots containing IO data.

The Application Process of IO devices may need parameter (e.g. the range of an analogue signal) from the assigned IO controller to provide the data from the Input Data object and/or for the Output Data object according to the requirements of the AP of the IO controller. These IO devices provide in the Context ASE an IO AR Parameter object or, for more flexibility, several IO AR Structured Parameter objects. The values for the attributes of these objects will be provided by the IO controller and assigned after validation from the AP of the IO device.

The functionality of an IO device application implies the necessary parameterization objects (specific Record Data objects) which have to be conveyed at the start-up phase of the system from the IO controller to the assigned IO devices via the IO AR. The application of the IO device has to check the parameterization data sent by the IO controller.

NOTE In comparison to the IO Data ASE, where all the actual available slots and subslots with there attributes are stored, the Context ASE contains only the configured or used slots and subslots after a successful connection establishment.

8.3.6.2 Context class specification

8.3.6.2.1 Template

This class specifies the object for the parameter data that are related to the IO devices itself and/or to the slots/subslots of the IO device.

The IO Device context object is described by the following template:

ASE:	**Context ASE**		
CLASS:	**Context**		
CLASS ID:	**not used**		
PARENT CLASS:	**TOP**		
ATTRIBUTES:			
1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	List of APs
2.1	(m)	Attribute:	API
2.2	(m)	Attribute:	Real Identification

2.2.1	(m)	Attribute:	List of Slots
2.2.1.1	(m)	Attribute:	Slot Number
2.2.1.2	(m)	Attribute:	Module Ident Number
2.2.1.3	(m)	Attribute:	Module Properties
2.2.1.4	(m)	Attribute:	List of Subslots
2.2.1.4.1	(m)	Attribute:	Subslot Number
2.2.1.4.2	(m)	Attribute:	Submodule Ident Number
2.2.1.4.3	(m)	Attribute:	Submodule Properties
2.2.1.4.3.1	(m)	Attribute:	Type
2.2.1.4.3.2	(m)	Attribute:	Shared Input
2.2.1.4.4	(m)	Attribute:	List of Data Descriptions
2.2.1.4.4.1	(m)	Attribute:	Data Direction
2.2.1.4.4.2	(m)	Attribute:	Submodule Data Length
2.2.1.4.4.3	(m)	Attribute:	Length IOPS
2.2.1.4.4.4	(m)	Attribute:	Length IOCS
2.3	(m)	Attribute:	Expected Identification
2.3.1	(m)	Attribute:	List of ARs
2.3.1.1	(m)	Attribute:	AREP
2.3.1.2	(o)	Attribute:	Sync State
2.3.1.3	(m)	Attribute:	List of CRs
2.3.1.3.1	(m)	Attribute:	CREP
2.3.1.3.2	(m)	Attribute:	Multicast Provider State
2.3.1.4	(m)	Attribute:	List of APIs
2.3.1.4.1	(m)	Attribute:	API
2.3.1.4.2	(m)	Attribute:	List of Slots
2.3.1.4.2.1	(m)	Attribute:	Slot Number
2.3.1.4.2.2	(m)	Attribute:	Module Ident Number
2.3.1.4.2.3	(m)	Attribute:	Module Properties
2.3.1.4.2.4	(m)	Attribute:	Module State
2.3.1.4.2.5	(m)	Attribute:	List of Subslots
2.3.1.4.2.5.1	(m)	Attribute:	Subslot Number
2.3.1.4.2.5.2	(m)	Attribute:	Submodule Ident Number
2.3.1.4.2.5.3	(m)	Attribute:	Submodule Properties
2.3.1.4.2.5.3.1	(m)	Attribute:	Type
2.3.1.4.2.5.3.2	(m)	Attribute:	Shared Input
2.3.1.4.2.5.4	(m)	Attribute:	Submodule State
2.3.1.4.2.5.4.1	(m)	Attribute:	Format Indicator
2.3.1.4.2.5.4.2	(c)	Constraint:	Format Indicator = V_1_0
2.3.1.4.2.5.4.2.1	(m)	Attribute:	Detail
2.3.1.4.2.5.4.3	(c)	Constraint:	Format Indicator <> V_1_0
2.3.1.4.2.5.4.3.1	(m)	Attribute:	Add Info
2.3.1.4.2.5.4.3.2	(m)	Attribute:	Qualifier Info
2.3.1.4.2.5.4.3.3	(m)	Attribute:	Maintenance Required
2.3.1.4.2.5.4.3.4	(m)	Attribute:	Maintenance Demanded
2.3.1.4.2.5.4.3.5	(m)	Attribute:	Diag Info
2.3.1.4.2.5.4.3.6	(m)	Attribute:	AR Info
2.3.1.4.2.5.4.3.7	(m)	Attribute:	Ident Info
2.4	(m)	Attribute:	I&M Data
2.4.1	(m)	Attribute:	List of Slots
2.4.1.1	(m)	Attribute:	Slot Number
2.4.1.4	(m)	Attribute:	List of Subslots
2.4.1.4.1	(m)	Attribute:	Subslot Number
2.4.1.4.2	(m)	Attribute:	I&M0 Data

2.4.1.4.2.1	(m)	Attribute:	Vendor ID
2.4.1.4.2.2	(m)	Attribute:	Order ID
2.4.1.4.2.3	(m)	Attribute:	Serial Number
2.4.1.4.2.4	(m)	Attribute:	Hardware Revision
2.4.1.4.2.5	(m)	Attribute:	Software Revision
2.4.1.4.2.5.1	(m)	Attribute:	Type Recognition
2.4.1.4.2.5.2	(m)	Attribute:	Functional Enhancement
2.4.1.4.2.5.3	(m)	Attribute:	Bug Fix
2.4.1.4.2.5.4	(m)	Attribute:	Minor Change
2.4.1.4.2.6	(m)	Attribute:	Revision Counter
2.4.1.4.2.7	(m)	Attribute:	Profile ID
2.4.1.4.2.8	(m)	Attribute:	Profile Specific Type
2.4.1.4.2.9	(m)	Attribute:	I&M Version
2.4.1.4.2.10	(m)	Attribute:	I&M Supported
2.4.1.4.3	(o)	Attribute:	I&M1 Data
2.4.1.4.3.1	(m)	Attribute:	Function
2.4.1.4.3.2	(m)	Attribute:	Location
2.4.1.4.4	(o)	Attribute:	I&M2 Data
2.4.1.4.4.1	(m)	Attribute:	Installation Date
2.4.1.4.5	(o)	Attribute:	I&M3 Data
2.4.1.4.5.1	(m)	Attribute:	Description
2.4.1.4.6	(o)	Attribute:	I&M4 Data
2.4.1.4.6.1	(m)	Attribute:	Signature
2.5	(m)	Attribute:	I&M0 Filter Data
2.5.1	(m)	Attribute:	I&M0 Carrier Data
2.5.1.1	(m)	Attribute:	List of APs
2.5.1.1.1	(m)	Attribute:	API
2.5.1.1.2	(m)	Attribute:	List of Slots
2.5.1.1.2.1	(m)	Attribute:	Slot Number
2.5.1.1.2.2	(m)	Attribute:	Module Ident Number
2.5.1.1.2.3	(m)	Attribute:	List of Subslots
2.5.1.1.2.3.1	(m)	Attribute:	Subslot Number
2.5.1.1.2.3.2	(m)	Attribute:	Submodule Ident Number
2.5.2	(o)	Attribute:	Module Representative Data
2.5.2.1	(m)	Attribute:	List of APs
2.5.2.1.1	(m)	Attribute:	API
2.5.2.1.2	(m)	Attribute:	List of Slots
2.5.2.1.2.1	(m)	Attribute:	Slot Number
2.5.2.1.2.2	(m)	Attribute:	Module Ident Number
2.5.2.1.2.3	(m)	Attribute:	List of Subslots
2.5.2.1.2.3.1	(m)	Attribute:	Subslot Number
2.5.2.1.2.3.2	(m)	Attribute:	Submodule Ident Number
2.5.3	(o)	Attribute:	Device Representative Data
2.5.3.1	(m)	Attribute:	List of APs
2.5.3.1.1	(m)	Attribute:	API
2.5.3.1.2	(m)	Attribute:	List of Slots
2.5.3.1.2.1	(m)	Attribute:	Slot Number
2.5.3.1.2.2	(m)	Attribute:	Module Ident Number
2.5.3.1.2.3	(m)	Attribute:	List of Subslots
2.5.3.1.2.3.1	(m)	Attribute:	Subslot Number
2.5.3.1.2.3.2	(m)	Attribute:	Submodule Ident Number

SERVICES:

1	(m)	OpsService:	Connect

2	(m)	OpsService:	Connect Device Access
3	(m)	OpsService:	Release
4	(m)	OpsService:	End Of Parameter
5	(m)	OpsService:	Application Ready
6	(m)	OpsService:	Read Expected Identification Data
7	(m)	OpsService:	Read Real Identification Data
8	(m)	OpsService:	Read Identification Difference
9	(m)	OpsService:	Abort
10	(o)	OpsService:	Ready For Companion
11	(m)	OpsService:	Read API Data
12	(m)	OpsService:	Read I&M0 Filter Data
13	(m)	OpsService:	Read I&M0 Data
14	(o)	OpsService:	Write I&M1 Data
15	(o)	OpsService:	Read I&M1 Data
16	(o)	OpsService:	Write I&M2 Data
17	(o)	OpsService:	Read I&M2 Data
18	(o)	OpsService:	Write I&M3 Data
19	(o)	OpsService:	Read I&M3 Data
20	(o)	OpsService:	Write I&M4 Data
21	(o)	OpsService:	Read I&M4 Data

8.3.6.2.2 Attributes

Implicit
The attribute Implicit indicates that the User Parameter object is implicitly addressed by the services.

List of APs
One API is composed of the following list elements:

API
This attribute defines the number of the application process.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

Real Identification
This attribute consists of the following attributes:

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This attribute defines to which module the Real Identification Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 7FFF

Module Ident Number
This attribute defines the identification of the slot.

Attribute type: Unsigned32

Allowed values: 1 to 0xFFFFFFFF

NOTE 1 The attribute value is manufacturer specific.

Module Properties
This attribute is reserved for future use.

Module State
This attribute defines the state of the module and shall be used for the connect service and the application ready service.

Attribute type: Unsigned16

Allowed values: Shall be set according to the Table 288.

Table 288 – Module State

Value	Use
NO_MODULE	The slot exists and the module is not plugged.
WRONG_MODULE	The module ident number is wrong.
PROPER_MODULE	The slot exists and the module is okay but at least one submodule is locked, wrong or missing.
SUBSTITUTE	The slot exists and the module is not the same as requested - but compatible. The IO device was able to adapt by its own decision.
GOOD	There shall be no entry in the attribute Module Diff Block for this module in the connect.rsp service and the application ready.req service.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This attribute defines to which subslot the Real Identification Data Element belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

Submodule Ident Number
This attribute defines the identification of the subslot.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE 2 The attribute value is manufacturer specific.

Submodule Properties
This attribute consists of the following attributes.

Type
This attribute defines the type of the subslot.

Attribute type: Unsigned16

Allowed values: NO_IO, INPUT, OUTPUT, IO

Shared Input
This attribute defines the properties of the subslot. The value SHARED_INPUT shall only used if the attribute Submodule type contains the value INPUT or IO.

Attribute type: Unsigned8

Allowed values: SHARED_INPUT,
NOT_SHARED_INPUT

List of Data Descriptions
This attribute list consists of the following attributes.

Data Direction
The attribute defines the direction input or output.

Attribute type: Unsigned16

Allowed values: DIRECTION_INPUT, DIRECTON_OUTPUT

Submodule Data Length
The attribute Submodule Data Length defines the number of octets of the Data Element without counting the IOPS.

Attribute type: Unsigned16

Allowed values: 0 to 1 439

Length IOPS
The attribute Length IOPS defines the number of octets of the provider status.

Attribute type: Unsigned8

Allowed values: 1, (2 to 255 reserved)

Length IOCS
The attribute Length IOCS defines the number of octets of the consumer status.

Attribute type: Unsigned8

Allowed values: 1, (2 to 255 reserved)

Expected Identification
This attribute consists of the following attributes:

List of ARs
One AR is composed of the following list elements:

AREP
This parameter is the local identifier for the desired AR.

SYNC State
This attribute is the local synchronization status for the desired AR. It shall only be present if synchronization is used. The behavior shall be according to 8.3.7.

Attribute type: Unsigned16

Allowed values: SYNCHRONIZED, SYNC_NOT_AVAILABLE

List of CRs
One CR is composed of the following list elements:

CREP
This parameter is the local identifier for the desired multicast consumer CR.

Multicast Provider State
This parameter is the local status for the multicast consumer CR. The behavior shall be according to 8.3.7.

Attribute type: Unsigned16

Allowed values: UP_AND_RUNNING,
MULTICAST_PROVIDER_NOT_AVAILABLE

List of APIs
One API is composed of the following list elements:

API
This attribute defines the number of the application process.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

List of Slots
One Slot is composed of the following list elements:

Slot Number
This attribute defines to which module the IO Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Module Ident Number
This attribute defines the identification of the slot.

Attribute type: Unsigned32

Allowed values: 1 to 0xFFFFFFFF

NOTE 3 The attribute value is manufacturer specific.

Module Properties
This attribute is reserved for further use.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This attribute defines to which subslot the IO Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF, for Slot Number 0 additionally 0x8000 to 0x8FFF are allowed

Submodule Ident Number
This attribute defines the identification of the subslot.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE 4 The attribute value is manufacturer specific.

Submodule Properties
This attribute consists of the following attributes.

Type
This attribute defines the type of the subslot.

Attribute type: Unsigned16

Allowed values: NO_IO, INPUT, OUTPUT, IO

Table 289 shows the dependencies that shall be applied in conjunction with selected CR attribute CR type.

Table 289 – Usage with respect to CR type

Submodule type	Usage with CR type
NO_IO	INPUT CR, OUTPUT CR
INPUT	INPUT CR, OUTPUT CR, MULTICAST PROVIDER CR
OUTPUT	INPUT CR, OUTPUT CR, MULTICAST CONSUMER CR
IO	INPUT CR, OUTPUT CR, MULTICAST PROVIDER CR (Input part only), MULTICAST CONSUMER CR (Output part only)

Shared Input
This attribute defines the properties of the subslot. The value SHARED_INPUT shall only used if the attribute Submodule type contains the value INPUT or IO.

Attribute type: Unsigned8

Allowed values: SHARED_INPUT,
NOT_SHARED_INPUT

Submodule State
This attribute consists of the following attributes:

Format Indicator
This attribute indicates the format of Submodule State

Detail
This attribute contains the value for the submodule state compatible to version 1.0.

Shall be set according to the Table 290.

Table 290 – Detail

Value	Meaning
NO_SUBMODULE	The subslot exists and is empty.
WRONG_SUBMODULE	The subslot exists and an incompatible submodule is plugged.
LOCKED_BY_IO_CONTROLLER	The subslot exists and is locked by another IO controller.
APPLICATION_READY_PENDING	The subslot exists and a parameter fault was detected or the application of the subslot is not operating
SUBSTITUTE	The subslot exists and the submodule is not the same as requested – but compatible and the IO device was able to adapt In this case the IO device has to adapt e.g. to new input or output length
GOOD	The subslot exists and the expected submodule is plugged.

NOTE 5 The field Submodule State is only responded if there is a difference between expected and real configuration data. Submodule State "GOOD" is not defined because for such submodules no status is reported within the response.

Add Info
This attribute contains a value for the IO supervisor in case the takeover was not allowed.

Allowed Values: NO_ADD_INFO, TAKEOVER_NOT_ALLOWED

Qualifier Info
This attribute signals the presence of qualified diagnosis information when application ready is generated.

Allowed Values: NO_QUALIFIED_INFO, QUALIFIED_INFO

Maintenance Required
This attribute contains a value MAINTENANCE_REQUIRED if at least one Channel of the Submodule requires maintenance.

Allowed Values: MAINTENANCE_REQUIRED,
NO_MAINTENANCE_REQUIRED

Maintenance Demanded
This attribute contains a value MAINTENANCE_DEMANDED if at least one Channel of the Submodule demands maintenance.

Allowed Values: MAINTENANCE_DEMANDED,
NO_MAINTENANCE_DEMANDED

Diag Info
This attribute signals the presence of diagnosis information when application ready is generated.

Allowed Values: NO_DIAG, DIAG

AR Info
This attribute contains the submodule state in relation to the established AR.

It shall be set according to the Table 291.

Table 291 – ARInfo

Value	Meaning
OWN	This AR is owner of the submodule
APPLICATION_READY_PENDING	This AR is owner of the submodule but it is blocked. I.e. parameter checking pending
SUPERORDINATED_LOCKED	This AR is not owner of the submodule. It is blocked by superordinated means
LOCKED_BY_IO_CONTROLLER	This AR is not owner of the submodule. It is owned by an other IOAR
LOCKED_BY_IO_SUPERVISOR	This AR is not owner of the submodule. It is owned by an other IOSAR

Ident Info
This attribute contains the submodule state in respect to expected and real identification for an established AR.

It shall be set according to the Table 292.

Table 292 – Ident Info

Value	Meaning
OK	No difference
SUBSTITUTE	Different but compatible
WRONG	Incompatible
NO	Empty

I&M Data
This attribute consists of the following attributes:

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This attribute defines to which module the I&M Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 7FFF

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This attribute defines to which subslot the I&M Data Element belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

I&M0 Data
This attribute consists of the following attributes.

Vendor ID
This attribute defines the ID of the submodule's manufacturer as assigned by the PROFIBUS business office. 0x0000 is used in case a manufacturer does not want to handle an ID.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF are allowed

Order ID
This attribute defines the complete order number or at least a relevant part that allows unambiguous identification of the submodule within the manufacturer's order spectrum.

The attribute is filled as visible string. Unused octets shall be set to 0x20 (blank).

Attribute type: Octet String[20]

Serial Number
This attribute defines the serial number of the submodule. The serial number is a unique production number of the device manufacturer even for devices with the same hardware, software or firmware edition.

The attribute is filled as visible string. Unused octets shall be set to 0x20 (blank).

Attribute type: Octet String[16]

Hardware Revision
This attribute defines the hardware edition of the submodule.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF are allowed

Software Revision
This attribute defines the software or firmware edition of the submodule. The structure supports coarse and detailed differentiation that may be defined by the manufacturer as “Vx.y.z”. Any component that carries software shall present its software version even if the user may not be aware of it. V255.255.255 indicates the availability of profile specific information.

This attribute consists of the following attributes:

Type Recognition
The type recognition attribute forms the “**V**x.y.z” part of the software revision.

Attribute type: Octet String[1]

Preferred values:

“V” (= officially released version) or “R” (= Revision)

“P” = Prototype

“U” = Under Test (field test)

“T” = Test Device

Functional Enhancement
The functional enhancement attribute forms the “V**x**.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF are allowed

Bug Fix
The bug fix attribute forms the “Vx.**y**.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF are allowed

Minor Change
The minor change attribute forms the “Vx.y.**z**” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF are allowed

Revision Counter
This attribute defines the revision counter of the submodule.

A change of the revision counter marks a change of hardware or of its parameters. At production time the counter will be set to 0. This value is reserved for the first installation and the first increment. Thus, the counter increments from 1... 65 535, wrapping over back to 1 at the end.

The counter is incremented on submodule exchange or on any write access of parameters within the submodule and its activation.

To avoid unnecessary counter increments more tolerant and sophisticated criteria, meeting the minimum requirements of FDA (Food & Drug Administration) may be implemented manufacturer specifically (e.g. batch-ID, local parameter server, etc).

The revision counter within the representative of the whole modular device – usually assigned to slot 0 – shall be incremented if any module's revision counter is incremented.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF are allowed

Profile ID
This attribute defines the ID of the application profile the submodule is following as assigned by the PROFIBUS business office. Submodules not following any PROFIBUS application profile shall use 0xF600.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF are allowed

Profile Specific Type
In case a submodule follows a special application profile, this attribute offers information about the usage of its channels and/or sub devices. The submodule/channel information shall be according to the respective definitions of the application profile.

In case a submodule does not follow a special application profile, the attribute indicates the type of sensor or actuator.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF are allowed

I&M Version
This attribute indicates the implemented version of the I&M functions.

Octet 1 (MSB) contains the major version number, e.g. 1 of version 1.0; octet 2 (LSB) contains the minor version number, e.g. 0 of version 1.0.

Attribute type: Octet String[2]

I&M Supported
This attribute indicates the availability of I&M records. Bit 0 indicates the availability of Profile specific I&M, Bit 1 the availability of I&M1, Bit 2 the availability of I&M2, … and Bit 15 the availabilty of I&M15.

Attribute type: Bit Sequence V2

I&M1 Data
This optional attribute consists of the following attributes.

Function
This attribute defines a unique label is necessary for the identification of the submodule's function or task. This may be a standard symbolic tag ("AKZ") out of a list or any other type of label defined by a configuration tool. Unused characters shall be set to 0x20 (blank).

Attribute type: Octet String[32]

Default value: Filled with 0x20 (blank)

Location
This attribute defines a unique label is necessary for the identification of the submodule's location. This may be a standard symbolic tag ("OKZ") out of a list or any other type of label defined by a configuration tool. Unused characters shall be set to 0x20 (blank).

Attribute type: Octet String[22]

Default value: Filled with 0x20 (blank)

I&M2 Data
This optional attribute consists of the following attribute.

Installation Date
This attribute indicates the date of installation or commissioning of a device or module. The content is a fixed date format according to ISO 8601: YYYY-MM-DD, where YYYY is the year in the usual Gregorian calendar, MM is the month of the year between 01 (January) and 12 (December), and DD is the day of the month between 01 and 31.

Time information shall be added: hh:mm, where hh is the hour of the day (0 - 23) and mm the minutes of the hour (0 - 59). Time and date shall be separated by a blank character.

For example, the fourth day of February in the year 1995 is written in the standard notation as:

1995-02-04 16:23

Unused characters shall be set to 0x20 (blank).

Attribute type: Octet String[16]

Default value: Filled with 0x20 (blank)

I&M3 Data
This optional attribute consists of the following attributes.

Descriptor
This attribute allows storing any individual additional information and annotation.

Unused characters shall be set to 0x20 (blank).

Attribute type: Octet String[54]

Default value: Filled with 0x20 (blank)

I&M4 Data
This optional attribute consists of the following attributes.

Signature
This attribute allows allows parameterization tools to store a "security" code as a reference for a particular parameterization session and audit trail tools to retrieve the code for integrity checks. Together with the "asset identification" consisting of Vendor ID, Order ID, and Serial Number the Signature allows unambiguousness of the session data.

Unused characters shall be set to 0x20 (blank).

Attribute type: Octet String[54]

Default value: Filled with 0x20 (blank)

I&M Filter Data
This attribute consists of the following attributes:

I&M Carrier Data

This attribute indicates all submodules being a carrirer of distinct I&M data. It consists of the following attributes.

List of APs
One API is composed of the following list elements:

API
This attribute defines the number of the application process.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This attribute defines to which module the I&M Filter Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 7FFF

Module Ident Number
This attribute defines the identification of the slot.

Attribute type: Unsigned32

Allowed values: 1 to 0xFFFFFFFF

NOTE 6 The attribute value is manufacturer specific.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This attribute defines to which subslot the I&M Filter Data Element belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

Submodule Ident Number
This attribute defines the identification of the subslot.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE 7 The attribute value is manufacturer specific.

Module Representative Data

This optional attribute indicates the submodules acting as module representative and thus carrying the module's I&M data. It consists of the following attributes.

NOTE 8 The attribute indicates at most one submodule per module acting as module representative.

List of APs
One API is composed of the following list elements:

API
This attribute defines the number of the application process.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This attribute defines to which module the I&M Filter Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 7FFF

Module Ident Number
This attribute defines the identification of the slot.

Attribute type: Unsigned32

Allowed values: 1 to 0xFFFFFFFF

NOTE 9 The attribute value is manufacturer specific.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This attribute defines to which subslot the I&M Filter Data Element belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

Submodule Ident Number
This attribute defines the identification of the subslot.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE 10 The attribute value is manufacturer specific.

Device Representative Data

This optional attribute indicates the submodule acting as device representative and thus carrying the device's I&M data. It consists of the following attributes.

NOTE 11 The attribute indicates at most one submodule acting as device representative.

List of APs
One API is composed of the following list elements:

API
This attribute defines the number of the application process.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This attribute defines to which module the I&M Filter Data Element belongs.

Attribute type: Unsigned16

Allowed values: 0 to 7FFF

Module Ident Number
This attribute defines the identification of the slot.

Attribute type: Unsigned32

Allowed values: 1 to 0xFFFFFFFF

NOTE 12 The attribute value is manufacturer specific.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This attribute defines to which subslot the I&M Filter Data Element belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

Submodule Ident Number
This attribute defines the identification of the subslot.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE 13 The attribute value is manufacturer specific.

8.3.6.2.3 Invocation of the Context object

Each IO Device shall invoke one IO Device Context object.

8.3.6.3 Context service specification

8.3.6.3.1 Connect

The Connect service is used to establish an IO AR or Supervisor AR. The application process shall check the service parameter of the request/indication with the local parameter in its ASE

objects (Real Identification attributes). In case of a service parameter mismatch (e.g. submodules are already used by other IO ARs) with the object attributes the application process shall respond the mismatching modules within response. The client may take the appropriate actions. If the establishment phase was successfully the service parameter are stored as attribute values in the Expected Identification attribute group of the IO Device Context object.

Table 293 shows the parameter of the service.

NOTE 1 The service parameter IP Address and Object UUID are necessary to address the destination device. The underlying model assumes that this information is provided by the AR information, which has to be loaded before on behalf of the client application. On server side this information is also handled inside the application layer and does not appear as an explicit service parameter. Local implementation may pass this service parameter to the application process. The same model applies to Read services over the implicit AR.

The application of the server receiving the Connect indication service primitive shall check the service parameter. Especially, the parameter Module Ident Number, Submodule Ident Number, Submodule Data Length, IOCS Length, and IOPS Length shall only be checked by the user application.

Table 293 – Connect

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
AR Parameter Block	M	M(=)		
AR type	M	M(=)		
AR UUID	M	M(=)		
Session Key	M	M(=)		
CM Initiator Mac Add	M	M(=)		
CM Initiator Object UUID	M	M(=)		
AR Properties	M	M(=)		
CM Initiator Activity Timeout Factor	M	M(=)		
Initiator UDP RT Port	M	M(=)		
CM Initiator Station Name	M	M(=)		
List of IO CR Parameter Blocks	M	M(=)		
IO CR type	M	M(=)		
IO CR Reference	M	M(=)		
LT Field	M	M(=)		
IO CR Properties	M	M(=)		
C_SDU Length	M	M(=)		
Frame ID	M	M(=)		
Send Clock Factor	M	M(=)		
Reduction Ratio	M	M(=)		
Phase	M	M(=)		
Sequence	M	M(=)		
Frame Send Offset	M	M(=)		
Watchdog Factor	M	M(=)		
Data Hold Factor	M	M(=)		
IO CR Tag Header	M	M(=)		
IO CR Multicast MAC Add	M	M(=)		
List Of APIs	M	M(=)		
API	M	M(=)		
List of Related IO Data Objects	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
IO Data Object Frame Offset	M	M(=)		
List of Related IOCS	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
IOCS Frame Offset	M	M(=)		
Alarm CR Parameter Block	M	M(=)		
Alarm CR type	M	M(=)		
LT Field	M	M(=)		

Parameter name	Req	Ind	Rsp	Cnf
Alarm CR Properties	M	M(=)		
RTA Timeout Factor	M	M(=)		
RTA Retries	M	M(=)		
Local Alarm Reference	M	M(=)		
Max Alarm Data Length	M	M(=)		
Alarm CR Tag Header High	M	M(=)		
Alarm CR Tag Header Low	M	M(=)		
List of Expected Submodule Blocks	U	U(=)		
List of Related APIs	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Module Ident Number	M	M(=)		
Module Properties	M	M(=)		
List of Submodules	M	M(=)		
Subslot Number	M	M(=)		
Submodule Ident Number	M	M(=)		
Submodule Properties	M	M(=)		
List of Data Descriptions	M	M(=)		
Data Description	M	M(=)		
Submodule Data Length	M	M(=)		
Length IOPS	M	M(=)		
Length IOCS	M	M(=)		
Parameter Server Block	U	U(=)		
Parameter Server Object UUID	M	M(=)		
Parameter Server Properties	M	M(=)		
CM Initiator Activity Timeout Factor	M	M(=)		
Parameter Server Station Name	M	M(=)		
List of Multicast CR Blocks	U	U(=)		
IOCR Reference	M	M(=)		
Address Resolution Properties	M	M(=)		
MCI Timeout Factor	M	M(=)		
Provider Station Name	M	M(=)		
AR RPC Block	U	U(=)		
Initiator RPC Server Port	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
AR Response Block			M	M(=)
AR type			M	M(=)
AR UUID			M	M(=)
Session Key			M	M(=)
CM Responder MAC Add			M	M(=)
Responder UDP RT Port			M	M(=)
List of IO CR Response Blocks			M	M(=)
IOCR type			M	M(=)
IOCR Reference			M	M(=)
Frame ID			M	M(=)
Alarm CR Response Block			M	M(=)
Alarm CR type			M	M(=)
Local Alarm Reference			M	M(=)
Max Alarm Data Length			M	M(=)
Module Diff Block			U	U(=)
List of APIs			M	M(=)
API			M	M(=)
List of Modules			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
Module State			M	M(=)
List of Submodules			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Submodule State			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
AR RPC Block			U	U(=)
Responder RPC Server Port			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

AR Parameter Block
The parameter contains of the following attributes:

AR type
This parameter contains the value for the equivalent attribute of the ARL class specification.

AR UUID
This parameter contains the value for the equivalent attribute of the ARL class specification.

Session Key
The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE 2 The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Attribute type: Unsigned16

CM Initiator MAC Add
This parameter contains the value for the equivalent attribute of the ARL class specification.

IO Initiator Object UUID
This parameter contains the value for the equivalent attribute of the ARL class specification.

AR Properties
This parameter contains the value for the equivalent attribute of the ARL class specification.

CM Initiator Activity Timeout Factor
This parameter contains the value for the equivalent attribute of the ARL class specification.

Initiator UDP RT Port
This parameter contains the value for the equivalent attribute of the ARL class specification.

CM Initiator Station Name
This parameter contains the value for the equivalent attribute of the ARL class specification.

List of IO CR Parameter Blocks
The parameter list contains the following attributes:

IO CR type
This parameter contains the value for the equivalent attribute of the CRL class specification.

IO CR Reference
This parameter contains the value to identify the CR within this service.

NOTE 3 It is used within the context of the connect service to link the IO Data elements with its CRs. In the local context of the IO device it is transferred into the CREP.

LT Field
This parameter contains the value for the equivalent attribute of the CRL class specification.

IO CR Properties
This parameter contains the value for the equivalent attribute of the CRL class specification.

C_SDU Length
This parameter contains the value for the equivalent attribute of the CRL class specification.

Frame ID
This parameter contains the value for the equivalent attribute of the CRL class specification.

Send Clock Factor
This parameter contains the value for the equivalent attribute of the CRL class specification.

Reduction Ratio
This parameter contains the value for the equivalent attribute of the CRL class specification.

Phase
This parameter contains the value for the equivalent attribute of the CRL class specification.

Sequence
This parameter contains the value for the equivalent attribute of the CRL class specification.

Frame Send Offset
This parameter contains the value for the equivalent attribute of the CRL class specification.

Watchdog Factor
This parameter contains the value for the equivalent attribute of the CRL class specification.

Data Hold Factor
This parameter contains the value for the equivalent attribute of the CRL class specification.

IO CR Tag Header
This parameter contains the value for the equivalent attribute of the CRL class specification.

IO CR Multicast MAC Add
This parameter contains the value for the equivalent attribute of the CRL class specification.

List Of APIs
This parameter contains the value for the equivalent attribute of the CRL class specification.

API
This parameter contains the value for the equivalent attribute of the CRL class specification.

List of Related IO Data Objects
The parameter list shall always be present. It contains the slots and subslots where read and write access is allowed. The application process of the server shall respond to write services addressed to objects beyond this negotiated address space with a negative response. Read access is always allowed to other objects. The following parameter shall be checked with the corresponding attributes of the IO Data object and the result in case of a positive response shall be stored as

attribute values of the Context object. The parameter consists of the following sub-parameter:

Slot Number
This parameter contains the value for the equivalent attribute of the Context ASE.

Sublot Number
This parameter contains the value for the equivalent attribute of the Context ASE.

IO Data Object Frame Offset
This parameter contains the value for the equivalent attribute of the Context ASE.

List of Related IOCS
This parameter contains the value for the equivalent attribute of the Context ASE.

Slot Number
This parameter contains the value for the equivalent attribute of the Context ASE.

Sublot Number
This parameter contains the value for the equivalent attribute of the Context ASE.

IOCS Frame Offset
This parameter contains the value for the equivalent attribute of the Context ASE.

Alarm CR Parameter Block
This parameter consists of the following attributes:

Alarm CR type
This parameter contains the value for the equivalent attribute of the CRL class specification.

LT Field
This parameter contains the value for the equivalent attribute of the CRL class specification.

Alarm CR Properties
This parameter contains the value for the equivalent attribute of the CRL class specification.

RTA Timeout Factor
This parameter contains the value for the equivalent attribute of the CRL class specification.

RTA Retries
This parameter contains the value for the equivalent attribute of the CRL class specification.

Local Alarm Reference
This parameter contains the value for the equivalent attribute of the CRL class specification.

Max Alarm Data Length
This parameter contains the value for the equivalent attribute of the CRL class specification.

Alarm CR Tag Header High
This parameter contains the value for the equivalent attribute of the CRL class specification.

Alarm CR Tag Header Low
This parameter contains the value for the equivalent attribute of the CRL class specification.

List of Expected Submodule Blocks
This parameter list consists of the following parameter:

NOTE 5 The list includes only “real” submodules but does not include “empty” slots.

List Of Related APIs
This parameter contains the value for the equivalent attribute of the CRL class specification.

API
This parameter contains the value for the equivalent attribute of the CRL class specification.

Slot Number
This parameter contains the value for the equivalent attribute of the Context ASE.

Module Ident Number
This parameter contains the value for the equivalent attribute of the Context ASE.

Module Properties
This parameter contains the value for the equivalent attribute of the Context ASE.

List of Submodules
This list consists of the following parameter:

Subslot Number
This parameter contains the value for the equivalent attribute of the Context ASE.

Submodule Ident Number
This parameter contains the value for the equivalent attribute of the Context ASE.

Submodule Properties
This parameter contains the value for the equivalent attribute of the Context ASE.

List of Data Descriptions
This list consists of the following attributes:

Data Description
This parameter contains the value for the equivalent attribute of the Context ASE.

Submodule Data Length
This parameter contains the value for the equivalent attribute of the Context ASE.

Length IOPS
This parameter contains the value for the equivalent attribute of the Context ASE.

Length IOCS
This parameter contains the value for the equivalent attribute of the Context ASE.

Parameter Server Block
This parameter consists of the following attributes:

Parameter Server Object UUID
This parameter contains the value for the equivalent attribute of the ASE.

Parameter Server Properties
This parameter contains the value for the equivalent attribute of the ASE.

CM Initiator Activity Timeout Factor
This parameter contains the value for the equivalent attribute of the ASE.

Parameter Server Station Name
This parameter contains the value for the equivalent attribute of the ASE.

List of Multicast CR Block
This parameter consists of the following attributes:

IOCR Reference
This parameter contains the value for the equivalent attribute of the AR ASE.

Address Resolution Properties
This parameter contains the value for the equivalent attribute of the AR ASE.

MCI Timeout Factor
This parameter contains the value for the equivalent attribute of the AR ASE.

Provider Station Name
This parameter contains the value for the equivalent attribute of the AR ASE.

AR RPC Block
This optional parameter shall be used by the IO device to address the Initiator RPC Server Port instead of the port of the endpoint mapper. It consists of the following attributes:

Initiator RPC Server Port
This parameter contains the value for the equivalent attribute of the ASE.

Result(+)
This parameter indicates that the service request succeeded. It shall only contain IO Data Response Block parameter for entries that differ from the requested parameter values.

AR Response Block
The selectable parameter contains of the following attributes:

AR type
This parameter contains the value for the equivalent attribute of the ARL class specification.

AR UUID
This parameter contains the value for the equivalent attribute of the ARL class specification.

Session Key
The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Attribute type: Unsigned16

CM Responder MAC Add
This parameter contains the value for the equivalent attribute of the ARL class specification.

Responder UDP RT Port
This parameter contains the value for the equivalent attribute of the ARL class specification.

List of IO CR Response Blocks
The parameter list contains the following attributes:

IOCR type
This parameter contains the value for the equivalent attribute of the CRL class specification.

IOCR Reference
This parameter contains the value mirrored from the request.

Frame ID
This parameter contains the value for the equivalent attribute of the CRL class specification.

Alarm CR Response Block
This parameter consists of the following attributes:

Alarm CR type
This parameter contains the value for the equivalent attribute of the CRL class specification.

Local Alarm Reference
This parameter contains the value for the equivalent attribute of the CRL class specification.

Max Alarm Data Length
This parameter contains the value for the equivalent attribute of the CRL class specification.

Module Diff Block
The parameter consists of the following sub-parameter for values which differ from the requested ones:

List of APIs
This parameter contains the APIs which differ from the requested configuration.

API
This parameter contains the value for the equivalent attribute of the ASE.

List of Modules
This parameter contains the modules which differ from the requested configuration.

Slot Number
This parameter contains the value for the equivalent attribute of the ASE.

Module Ident Number
This parameter contains the value for the equivalent attribute of the ASE.

Module State
This parameter contains the value for the equivalent attribute of the ASE.

List of Submodules
This parameter list consists of the following parameter:

Subslot Number
This parameter contains the value for the equivalent attribute of the ASE.

Submodule Ident Number
This parameter contains the value for the equivalent attribute of the ASE.

Submodule State
This parameter contains the value for the equivalent attribute of the ASE.

AR RPC Block
This optional parameter shall be used by the initiator to address the Responder RPC Server Port instead of the port of the endpoint mapper. It consists of the following attributes:

Responder RPC Server Port
This parameter contains the value for the equivalent attribute of the ASE.

NOTE 6 It is necessary to query the endpoint mapper to get this port to avoid firewall problems.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter shall have the value PNIO.

Type: Unsigned8

Error code 1
The Error code 1 assumes one of the following values:
FAULTY_CONNECT_BLOCK.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

8.3.6.3.2 Connect Device Access

The Connect service is used to establish a Supervisor AR for device access. The application process shall check the service parameter of the request/indication with the local parameter in its ASE objects (Real Identification attributes). The client may take the appropriate actions. If the establishment phase was successfully the service parameter are stored as attribute values in the Expected Identification attribute group of the IO Device Context object.

Table 294 shows the parameter of the service.

NOTE 1 The service parameter IP Address and Object UUID are necessary to address the destination device. The underlying model assumes that this information is provided by the AR information, which has to be loaded before on behalf of the client application. On server side this information is also handled inside the application layer and does not appear as an explicit service parameter. Local implementation may pass this service parameter to the application process. The same model applies to Read services over the implicit AR.

The application of the server receiving the Connect indication service primitive shall check the service parameter.

Table 294 – Connect Device Access

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
AR Parameter Block	M	M(=)		
AR type	M	M(=)		
AR UUID	M	M(=)		
Session Key	M	M(=)		
CM Initiator Mac Add	M	M(=)		
CM Initiator Object UUID	M	M(=)		
AR Properties	M	M(=)		
CM Initiator Activity Timeout Factor	M	M(=)		
Initiator UDP RT Port	M	M(=)		
CM Initiator Station Name	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
AR Response Block			M	M(=)
AR type			M	M(=)
AR UUID			M	M(=)
Session Key			M	M(=)
CM Responder MAC Add			M	M(=)
Responder UDP RT Port			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

AR Parameter Block
The parameter contains of the following attributes:

AR type
This parameter contains the value for the equivalent attribute of the ARL class specification.

AR UUID
This parameter contains the value for the equivalent attribute of the ARL class specification.

Session Key
The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE 2 The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Attribute type: Unsigned16

CM Initiator MAC Add
This parameter contains the value for the equivalent attribute of the ARL class specification.

IO Initiator Object UUID
This parameter contains the value for the equivalent attribute of the ARL class specification.

AR Properties
This parameter contains the value for the equivalent attribute of the ARL class specification.

CM Initiator Activity Timeout Factor
This parameter contains the value for the equivalent attribute of the ARL class specification.

Initiator UDP RT Port
This parameter contains the value for the equivalent attribute of the ARL class specification.

CM Initiator Station Name
This parameter contains the value for the equivalent attribute of the ARL class specification.

Result(+)
This parameter indicates that the service request succeeded. It shall only contain IO Data Response Block parameter for entries that differ from the requested parameter values.

AR Response Block
The selectable parameter contains of the following attributes:

AR type
This parameter contains the value for the equivalent attribute of the ARL class specification.

AR UUID
This parameter contains the value for the equivalent attribute of the ARL class specification.

Session Key
The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Attribute type: Unsigned16

CM Responder MAC Add
This parameter contains the value for the equivalent attribute of the ARL class specification.

Responder UDP RT Port
This parameter contains the value for the equivalent attribute of the ARL class specification.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter shall have the value PNIO.

Type: Unsigned8

Error code 1
The Error code 1 assumes one of the following values:
FAULTY_CONNECT_BLOCK.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

8.3.6.3.3 Release

This service shall be used by an IO controller or IO supervisor to disconnect an IO or Supervisor AR.

Table 295 shows the parameter of the service.

Table 295 – Release

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Session Key	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Session Key			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Session Key
The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter shall contain the value PNIO.

Type: Unsigned8

Error code 1
The Error code 1 assumes one of the following values:
FAULTY_RELEASE_BLOCK

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

8.3.6.3.4 Abort

This service shall be used by an IO device to close an IO or Supervisor AR in case of a serious error when operation is not longer possible.

Table 296 shows the parameter of the service.

Table 296 – Abort

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
Result(+)			M

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Result(+)
This parameter indicates that the service request succeeded.

8.3.6.3.5 End Of Parameter

This service shall be used by the client (IO controller or IO Parameter server) to inform the IO device that all parameter are written by means of the Write service of the Record Data ASE.

Table 297 shows the parameter of the service.

Table 297 – End Of Parameter

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Session Key	M	M(=)		
Alarm Sequence Number	U	U(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Session Key			M	M(=)
Alarm Sequence Number			U	U(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Session Key
The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Alarm Sequence Number
This parameter shall only be used in conjunction with a plug or pull alarm. The value shall be taken from the Alarm Notification.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter shall have the value PNIO.

Type: Unsigned8

Error code 1
The Error code 1 assumes one of the following values:
FAULTY_END_OF_PARAMETER_BLOCK

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

8.3.6.3.6 Application Ready

By this service the User of an IO device indicates to the IO controller that the application is ready. This service shall only be used in conjunction with the IO AR.

Table 298 shows the parameters of the service.

Table 298 – Application Ready

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Session Key	M	M(=)		
Ready For Companion	U	U(=)		
Alarm Sequence Number	U	U(=)		
Module Diff Block	U	U(=)		
List of APIs	M	M(=)		
API	M	M(=)		
List of Modules	M	M(=)		
Slot Number	M	M(=)		
Module Ident Number	M	M(=)		
Module State	M	M(=)		
List of Submodules	M	M(=)		
Subslot Number	M	M(=)		
Submodule Ident Number	M	M(=)		
Submodule State	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Session Key			M	M(=)
Alarm Sequence Number			U	U(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Session Key
The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE 1 The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Ready For Companion
This parameter shall only be used for RT class 3 connection establishment. The value of the parameter Ready For Companion shall be set to TRUE if all prerequisites of RT class 3 operation of all related ports are fulfilled. Otherwise the value shall be set to FALSE.

Type: Boolean

NOTE 2 The service Ready For Companion is used to indicate fulfillment of prerequisites for RT class 3 operation at a later point of time.

Alarm Sequence Number
This parameter shall only be used in conjunction with a plug or pull alarm. The value shall be taken from the Alarm Notification.

Module Diff Block
This parameter shall only be if a difference exists. The parameter consists of the following sub-parameter for values which differ from the requested ones:

List of APIs
This parameter contains a list of APIs for the modules which differ from the requested configuration. A list element consists of the following parameter.

API
This parameter contains the value for the equivalent attribute of the ASE.

List of Modules
This parameter contains a list of modules which differ from the requested configuration. A list element consists of the following parameter.

Slot Number
This parameter contains the value for the equivalent attribute of the ASE.

Module Ident Number
This parameter contains the value for the equivalent attribute of the ASE.

Module State
This parameter contains the value for the equivalent attribute of the ASE.

List of Submodules
This parameter contains a list of submodules which differ from the requested configuration. A list element consists of the following parameter.

Subslot Number
This parameter contains the value for the equivalent attribute of the ASE.

Submodule Ident Number
This parameter contains the value for the equivalent attribute of the ASE.

Submodule State
This parameter contains the value for the equivalent attribute of the ASE.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter shall have the value PNIO.

Type: Unsigned8

Error code 1
The Error code 1 assumes one of the following values:
FAULTY_APPLICATION_READY_BLOCK

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

8.3.6.3.7 Ready For Companion

This service shall be used by the IO device to inform the IO controller that all prerequisits for RT class 3 operation of related ports are fulfilled if it was not indicated by means of Application Ready service.

Table 299 shows the parameter of the service.

Table 299 – Ready For Companion

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter shall have the value PNIO.

Type: Unsigned8

Error code 1
The Error code 1 assumes one of the following values:
FAULTY_CONTROL_BLOCK

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

8.3.6.3.8 Read Expected Identification

This confirmed service may be used to read the value of the attributes which contain the configured module and submodule identification related to all ARs of the device. The expected identification is configured by project planning. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 300 shows the parameters of the service.

NOTE 1 Using this service the data length may exceed the limit of the supported data length. In this case the service returns a negative response. In this case the slot specific service can be used to reduce the amount of data.

By means of the service parameter Target AR UUID, Slot, and Subslot the content of expected identification can be restricted to a specific AR, a specific Slot or a specific Subslot as a filter function. In case a user specific filter is selected the Read Expected Identification response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain Expected Identification Data if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 300 – Read Expected Identification

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Block Version Low = 0			S	S(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Block Version Low = 1			S	S(=)
List of APIs			M	M(=)
API			M	M(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This conditional parameter shall be used to address the desired API.

Target AR UUID
This parameter shall only be used to read AR specific expected identification information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE 2 It is used to read only diagnosis information the requested AR is connected to.

Slot Number
The parameter Slot Number is used to address only diagnosis information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used is used to address only diagnosis information for a specific slot/subslot. This service parameter shall only be used together with the service parameter Slot Number.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the Expected Identification Data that has to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

Block Version Low = 0
This parameter contains the structure identification of the following ASE objects. Block Version Low = 0 shall be used, if the IO device only support API 0.

List of Slots
This parameter is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value corresponding attribute of the ASE object.

List of Subslots
This parameter is composed of the following list elements:

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value corresponding attribute of the ASE object.

Block Version Low = 1
This parameter contains the structure identification of the following ASE objects. Block Version Low = 1 shall be used, if the IO device only support API 0.

List of APIs
This parameter is composed of the following list elements:

API
This parameter contains the value of the corresponding attribute of the ASE object.

List of Slots
This parameter is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value corresponding attribute of the ASE object.

List of Subslots
This parameter is composed of the following list elements:

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.9 Read Real Identification

This confirmed service may be used to read the value of the attributes which contain the module and submodule identification from the perspective of the IO device. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 301 shows the parameters of the service.

By means of the service parameter API, Target AR UUID, Slot, and Subslot the content of expected identification can be restricted to a specific AR, a specific Slot or a specific Subslot as a filter function. In case a user specific filter is selected the Read Real Identification response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

NOTE 1 If a module or submodule is pulled it's entry within the real identification is removed.

Implicit AR using Target AR UUID: This service does only contain Real Identification Data if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Implicit AR using API: This service does only contain Real Identification Data if the requested API exists. Otherwise a parameter error is responded.

Table 301 – Read Real Identification

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Block Version Low = 0			S	S(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Block Version Low = 1			S	S(=)
List of APIs			M	M(=)
API			M	M(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall only be used to read AR specific expected identification information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE 2 It is used to read only diagnosis information the requested AR is connected to.

Slot Number
The parameter Slot Number is used to address only diagnosis information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used is used to address only diagnosis information for a specific slot/subslot. This service parameter shall only be used together with the service parameter Slot Number.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the Real Identification Data that has to be read. The allowed length is from 0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

Block Version Low = 0
This parameter contains the structure identification of the following ASE objects. Block Version Low = 0 shall be used, if the IO device only support API 0.

List of Slots
This parameter is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value corresponding attribute of the ASE object.

List of Subslots
This parameter is composed of the following list elements:

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value corresponding attribute of the ASE object.

Block Version Low = 1
This parameter contains the structure identification of the following ASE objects. Block Version Low = 1 shall be used, if the IO device only support API 0.

List of APIs
This parameter is composed of the following list elements:

API
This parameter contains the value of the corresponding attribute of the ASE object.

List of Slots
This parameter is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value corresponding attribute of the ASE object.

List of Subslots
This parameter is composed of the following list elements:

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.10 Read Identification Difference

This confirmed service may be used to read the difference of the configured module and submodule identification related to the expected configuration. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 302 shows the parameters of the service.

By means of the service parameter Target AR UUID the content of identification difference is restricted to a specific AR.

Implicit AR using Target AR UUID: This service does only contain Identification Difference if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

If there exists no difference the Read Identification Difference service shall respond no data with Length zero.

Table 302 – Read Identification Difference

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Target AR UUID	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of APIs			U	U(=)
API			U	U(=)
List of Slots			U	U(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
Module State			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Submodule State			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Target AR UUID
This parameter shall only be used to read AR specific expected identification information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE 1 It is used to read only diagnosis information the requested AR is connected to.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the Expected Identification Data that has to be read. The allowed length is from 0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

List of APIs
This parameter consists of the following elements.

API
This parameter contains the value for the equivalent attribute of the ASE.

List of Slots
This parameter is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value corresponding attribute of the ASE object.

Module State
This parameter contains the value corresponding attribute of the ASE object.

List of Subslots
This parameter is composed of the following list elements:

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value corresponding attribute of the ASE object.

Submodule State
This parameter contains the value corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 2 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 3 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.11 Read AP Data

This confirmed service may be used to read the supported application profiles. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 303 shows the parameters of the service.

Table 303 – Read AP Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of APs			M	M(=)
API			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the API Data that has to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

List of APs
This parameter is composed of the following list elements:

API
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.12 Read I&M0 Filter Data

This confirmed service may be used to read the value of the attribute I&M0 Filter Data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 304 shows the parameters of the service.

The I&M0 Filter Data indicates all submodules being a carrier of distinct I&M0 Data, and optionally the submodules acting as module representative and acting as device representative.

Table 304 – Read I&M0 Filter Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
I&M0 Filter Data			M	M(=)
I&M0 Carrier Data			M	M(=)
List of APs			M	M(=)
API			M	M(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Module Represenative Data			O	O(=)
List of APs			M	M(=)
API			M	M(=)
List of Slots			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Device Representative Data			O	O(=)
List of APs			M	M(=)
API			M	M(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the API Data that has to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

I&M0 Filter Data
This attribute consists of the following attributes:

I&M0 Carrier Data

This attribute contains all submodules with discrete I&M0 data. Submodules with derived I&M0 data (e.g. virtual subsmodules like firmware function blocks) shall not be part of the I&M0 Carrier Data. It consists of the following attributes.

List of APs
This attribute is composed of the following list elements:

API
This parameter contains the value of the corresponding attribute of the ASE object.

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Representative Data

This optional attribute contains the module references and consists of the following attributes.

List of APs
This attribute is composed of the following list elements:

API
This parameter contains the value of the corresponding attribute of the ASE object.

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value of the corresponding attribute of the ASE object.

Device Representative Data

This optional attribute contains the device references and consists of the following attributes.

List of APs
This attribute is composed of the following list elements:

API
This parameter contains the value of the corresponding attribute of the ASE object.

List of Slots
This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number
This parameter contains the value of the corresponding attribute of the ASE object.

List of Subslots
This attribute consists of the following attributes.

Subslot Number
This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.13 Read I&M0 Data

This confirmed service may be used to read I&M0 Data. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 305 shows the parameters of the service.

Table 305 – Read I&M0 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
I&M0 Data			U	U(=)
Vendor ID			M	M(=)
Order ID			M	M(=)
Serial Number			M	M(=)
Hardware Revision			M	M(=)
Software Revision			M	M(=)
Revision Counter			M	M(=)
Profile ID			M	M(=)
Profile Specific Type			M	M(=)
I&M Version			M	M(=)
I&M Supported			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired I&M0 Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired I&M0 Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding

will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the I&M0 Data to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

Length
The parameter Length indicates the number of octets of the I&M0 Data read.

Special case (legacy submodules / modules):

The value zero, in conjunction with an entry in the I&M0 Carrier Data, indicates, that the submodule is not able to retrieve it's I&M0 data.

NOTE 1 In this case the engineering system may use locally generated I&M0 data.

I&M0 Data
This parameter consists of the following elements.

Vendor ID
This parameter contains the value of the corresponding attribute of the ASE object.

Order ID
This parameter contains the value of the corresponding attribute of the ASE object.

Serial Number
This parameter contains the value of the corresponding attribute of the ASE object.

Hardware Revision
This parameter contains the value of the corresponding attribute of the ASE object.

Software Revision
This parameter contains the value of the corresponding attribute of the ASE object.

Revision Counter
This parameter contains the value of the corresponding attribute of the ASE object.

Profile ID
This parameter contains the value of the corresponding attribute of the ASE object.

Profile Specific Type
This parameter contains the value of the corresponding attribute of the ASE object.

I&M Version
This parameter contains the value of the corresponding attribute of the ASE object.

I&M Supported
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 2 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 3 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.6.3.14 Write I&M1 Data

This confirmed service may be used to write the attributes of the I&M1 Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 306 shows the parameters of the service.

The service parameter “Multiple” may be used to convey multiple I&M1 Data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE 1 It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 306 – Write I&M1 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
I&M1 Data	M	M(=)		
Function	M	M(=)		
Location	M	M(=)		
Prm Flag		M		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Slot Number
The parameter Slot Number is used to address I&M1 information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used to address I&M1 information for a specific subslot.

Multiple
The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
This parameter indicates the number of octets of the I&M1 Data that is to be written. The allowed length is from 2^{0} to 2^{32}-256.

I&M1 Data
This parameter consists of the following elements.

Function
This parameter contains the value of the corresponding attribute of the ASE object.

Location
This parameter contains the value of the corresponding attribute of the ASE object.

Prm Flag
The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

NOTE 2 This parameter is used to control the persistent storage of data. It is utilized in fast start-up procedure.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.15 Read I&M1 Data

This confirmed service may be used to read I&M1 Data. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 307 shows the parameters of the service.

Table 307 – Read I&M1 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
I&M1 Data			U	U(=)
Function			M	M(=)
Location			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired I&M1 Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired I&M1 Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the I&M1 Data to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

Length
The parameter Length indicates the number of octets of the I&M1 Data read.

I&M1 Data
This parameter consists of the following elements.

Function
This parameter contains the value of the corresponding attribute of the ASE object.

Location
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.6.3.16 Write I&M2 Data

This confirmed service may be used to write the attributes of the I&M2 Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 308 shows the parameters of the service.

The service parameter "Multiple" may be used to convey multiple I&M2 Data within one APDU. The server shall mirror the value of "Multiple" within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE 1 It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 308 – Write I&M2 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
I&M2 Data	M	M(=)		
Installation Date	M	M(=)		
Prm Flag		M		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Slot Number
The parameter Slot Number is used to address I&M2 data for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used to address I&M2 data for a specific subslot.

Multiple
The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value

MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
This parameter indicates the number of octets of the I&M2 Data that is to be written. The allowed length is from 2^0 to 2^{32}-256.

I&M2 Data
This parameter consists of the following elements.

Installation Date
This parameter contains the value of the corresponding attribute of the ASE object.

Prm Flag
The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

NOTE 2 This parameter is used to control the persistent storage of data. It is utilized in fast start-up procedure.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.17 Read I&M2 Data

This confirmed service may be used to read I&M2 Data. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 309 shows the parameters of the service.

Table 309 – Read I&M2 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
I&M2 Data			U	U(=)
Installation Date			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired I&M2 Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired I&M2 Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the I&M2 Data to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

Length
The parameter Length indicates the number of octets of the I&M2 Data read.

I&M2 Data
This parameter consists of the following elements.

Installation Date
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.6.3.18 Write I&M3 Data

This confirmed service may be used to write the attributes of the I&M3 Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 310 shows the parameters of the service.

The service parameter “Multiple” may be used to convey multiple I&M3 Data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE 1 It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 310 – Write I&M3 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
I&M3 Data	M	M(=)		
Descriptor	M	M(=)		
Prm Flag		M		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Slot Number
The parameter Slot Number is used to address I&M3 data for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used to address I&M3 data for a specific subslot.

Multiple
The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
This parameter indicates the number of octets of the I&M3 Data that is to be written. The allowed length is from 2^0 to 2^{32}-256.

I&M3 Data
This parameter consists of the following elements.

Descriptor
This parameter contains the value of the corresponding attribute of the ASE object.

Prm Flag
The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

NOTE 2 This parameter is used to control the persistent storage of data. It is utilized in fast start-up procedure.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.19 Read I&M3 Data

This confirmed service may be used to read I&M3 Data. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 311 shows the parameters of the service.

Table 311 – Read I&M3 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
I&M3 Data			U	U(=)
Descriptor			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired I&M3 Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired I&M3 Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the I&M3 Data to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

Length
The parameter Length indicates the number of octets of the I&M3 Data read.

I&M3 Data
This parameter consists of the following elements.

Descriptor
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.6.3.20 Write I&M4 Data

This confirmed service may be used to write the attributes of the I&M4 Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 313 shows the parameters of the service.

The service parameter “Multiple” may be used to convey multiple I&M4 Data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE 1 It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 312 – Write I&M4 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
I&M4 Data	M	M(=)		
Signature	M	M(=)		
Prm Flag		M		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Slot Number
The parameter Slot Number is used to address I&M4 data for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used to address I&M4 data for a specific subslot.

Multiple
The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
This parameter indicates the number of octets of the I&M4 Data that is to be written. The allowed length is from 2^0 to 2^{32}-256.

I&M4 Data
This parameter consists of the following elements.

Signature
This parameter contains the value of the corresponding attribute of the ASE object.

Prm Flag
The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

NOTE 2 This parameter is used to control the persistent storage of data. It is utilized in fast start-up procedure.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 3 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 4 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.6.3.21 Read I&M4 Data

This confirmed service may be used to read I&M4 Data. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 313 shows the parameters of the service.

Table 313 – Read I&M4 Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
I&M4 Data			U	U(=)
Signature			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number
The parameter Slot Number is used in the destination device for addressing the desired I&M4 Data object in a specific slot (typically a module).

Subslot Number
The parameter Subslot Number is used in the destination device for addressing the desired I&M4 Data object in a specific subslot (typically a submodule).

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the I&M4 Data to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

Length
The parameter Length indicates the number of octets of the I&M4 Data read.

I&M4 Data
This parameter consists of the following elements.

Signature
This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 1 Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 2 Add Data 2 can be used by device vendors to transmit specific error messages.

8.3.6.4 Behavior of the Context object

The Context object contains the real identification data from the perspective of the IO device. The real identification is detected by local means before the IO device may communicate. This real identification may be changed during operation by a local means. It may effect adding or removing of slots with their subslots. In such a case, the real identification shall be updated and the IO controllers or IO supervisor connected via affected ARs shall be informed by means of an Alarm Notification service. The Alarm Types “Released”, “Controlled By Supervisor”, “Pull Alarm”, “Plug Alarm” or “Plug Wrong Submodule” shall be used to indicate such events.

An affected AR means an established relationship where at least one of the changed slot/subslot is part of the attribute Expected identification.

When receiving the Alarm Notification with Alarm type “Plug Alarm” the IO controller shall check if the new added slot/subslot needs additional parameter and convey this parameter to the IO device. The IO controller may also inform the local user.

When receiving the Alarm Notification with Alarm type “Pull Alarm” or “Plug Wrong Submodule” the IO controller may inform the local user and wait for further actions.

However, in the time between a Connect response and receiving the End Of Parameter indication, the real identification shall be kept consistent in any case. If “real” modules are removed within this time interval then it shall be reported afterwards by means of Application Ready. Furthermore, if “real” modules are plugged within this time interval then it shall be reported afterwards by means of Alarm Notifications. It is not allowed to respond inconsistent or not plausible Module Diff Blocks with the Connect response or Application Ready request service primitives.

NOTE The following example illustrates an inconsistent Module Diff Block. If a submodule state was reported in the Connect response it could not be missed in the following Application Ready request, even if it was plugged in between. However, a submodule can change to a bad state because of faulty submodule parameter.

8.3.6.5 Behavior of the attribute Submodule State

The attribute behavior of an IO device application is described in 8.4.6.

8.3.7 Isochronous Mode Application ASE

8.3.7.1 Overview

This ASE specifies the structure of the necessary parameters for isochronous mode of applications. The IsoM ASE provides a set of services used to read, and write their values. This ASE is an option and does not effect non-synchronous applications.

8.3.7.2 Isochronous Mode Application class specification

8.3.7.2.1 Template

An Isochronous Mode Application process object is described by the following template:

ASE: **Isochronous Mode Application ASE**
CLASS: **Isochronous Mode Application**
CLASS ID: **not used**
PARENT CLASS: **TOP**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	List of Modules
2.1	(m)	Attribute:	Slot Number
2.2	(m)	Attribute:	List of Submodules
2.2.1	(m)	Attribute:	Subslot Number
2.2.2	(m)	Attribute:	IsoM Data
2.2.2.1	(m)	Attribute:	Time Data Cycle
2.2.2.2	(m)	Attribute:	Time IO Input
2.2.2.3	(m)	Attribute:	Time IO Output
2.2.2.4	(m)	Attribute:	Time IO Input Valid
2.2.2.5	(m)	Attribute:	Time IO Output Valid
2.2.2.6	(m)	Attribute:	Controller Application Cycle Factor
2.2.3	(m)	Attribute:	Device Parameter
2.2.3.1	(m)	Attribute:	Time Data Cycle Base
2.2.3.2	(m)	Attribute:	Time IO Base
2.2.3.3	(m)	Attribute:	Time Data Cycle Min
2.2.3.4	(m)	Attribute:	Time Data Cycle Max
2.2.3.5	(m)	Attribute:	Time IO Input Min
2.2.3.6	(m)	Attribute:	Time IO Output Min

SERVICES:

1	(m)	OpsService:	Write IsoM Data
2	(m)	OpsService:	Read IsoM Data
3	(m)	OpsService:	SYNCH Event

8.3.7.2.2 Attributes

Implicit
The attribute Implicit indicates that the Isochronous Mode Application process object is implicitly addressed by the service.

List of Modules
This attribute list contains the following attributes:

Slot Number
This attribute defines to which module the Isochronous Mode Application attribute belongs.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Submodules
This attribute list contains the following attributes:

Subslot Number
This attribute defines to which subslot the Isochronous Mode Application attribute belongs.

Attribute type: Unsigned16

Allowed values: 1 to 0x7FFF

IsoM Data
This attribute consists of the following attributes:

Time Data Cycle (T_DC)
This attribute indicates the time factor for the application data cycle. This time includes all parts of the isochronous data cycle to convey inputs and outputs. The cycle length is a multiple of T_DC_BASE. The time for T_DC is calculated T_DC * T_DC_BASE * 31,25 µs.

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Time IO Input (T_IO_Input)
This attribute indicates the time factor to fetch the inputs for the IO device. The time base is value of T_IO_Base multiplied with 1 ns. The time for T_IO_Input is calculated T_IO_Input * T_IO_Base * 1 ns. The time (T_IO_Input) is related to the start of the Time Data Cycle and describes a time before the start of Time Data Cycle.

For T_IO_Input the following relation shall be applied:

(T_IO_InputValid+T_IO_InputMin) ≤ T_IO_Input ≤ T_DC

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Time IO Output (T_IO_Output)
This attribute indicates the time factor to set the outputs for the IO device. The time base is value of T_IO_Base multiplied with 1 ns. The time for T_IO_Output is calculated T_IO_Output * T_IO_Base * 1 ns. The time (T_IO_Output) is related to the start of the Time Data Cycle and describes a time after the start of Time Data Cycle.

For T_IO_Output the following relation shall be applied:

(T_IO_OutputValid + T_IO_OutputMin) ≤ T_IO_Output ≤ T_DC

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Time IO Input Valid (T_IO_InputValid)
This attribute indicates the point of time the new input data shall be availble to convey. This point of time is related to the start of the next cycle. The time is based on T_IO_Base.

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Time IO Output Valid (T_IO_OutputValid)
This attribute indicates the point of time the new output data are availble to process. This point of time is related to the start of T_DC. The time is based on T_IO_Base.

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Controller Application Cycle Factor (CACF)
This attribute indicates the IO Controller Application cycle time as a factor in multiples of T_DC. It determines the time the application of the IO Controller needs to process an isochronous application task completely. From isochronous point of view there is exactly one user task for each AR controlling the isochronous part of application. The synchronization point is determined by Send Clock within the selected Phase.

Attribute type: Unsigned16

Allowed values: 1 to 14

Device Parameter
This attribute consists of the following attributes.

Time Data Cycle Base (T_DC_Base)
This attribute contains the factor for the granularity of the adjustment of the data timeliness for the application cycle.

The time basis for T_DC_Base is 31,25 µs. The time for T_DC_Base is calculated T_DC_Base * 31,25 µs.

This field is part of the GSDML.

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Time IO Base (T_IO_Base)
This attribute contains the factor for the granularity of the adjustment of the input or output delay time.

The time basis for T_IO_Base is 1 ns. The time for T_IO_Base is calculated T_IO_Base * 1 ns.

This field is part of the GSDML.

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Time Data Cycle Min (T_DC_Min)
This attribute contains the factor for the smallest possible data cycle.

The time basis for T_DC_Min is the value of the attribute T_DC_Base. The time for T_DC_Min is calculated T_DC_Min * T_DC_Base * 31,25 µs.

This field is part of the GSDML.

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Time Data Cycle Max (T_DC_Max)
This attribute contains the factor for the longest possible data cycle.

The time basis for T_DC_Max is the value of the attribute T_DC_Base. The time for T_DC_Max is calculated T_DC_Max * T_DC_Base * 31,25 µs.

This field is part of the GSDML.

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Time IO Input Min (T_IO_InputMin)
This attribute contains the factor for the smallest possible input refresh delay. It may be the smallest possible time to copy or preprocess the input data. The time is related to the start of the application cycle.

The time basis for T_IO_InputMin is the value of the attribute T_IO_Base. The time for T_IO_InputMin is calculated T_IO_InputMin * T_IO_Base * 1 ns.

This field is part of the GSDML.

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Time IO Output Min (T_IO_OutputMin)
This attribute contains the factor for the smallest possible output refresh delay. It may be the smallest possible time to copy or preprocess the output data. The time is related to the end of the application cycle.

The time basis for T_IO_OutputMin is the value of the attribute T_IO_Base. The time for T_IO_OutputMin is calculated T_IO_OutputMin * T_IO_Base * 1 ns.

This field is part of the GSDML.

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Figure 91 shows the relation of the ASE attributes within the isochronous application model.

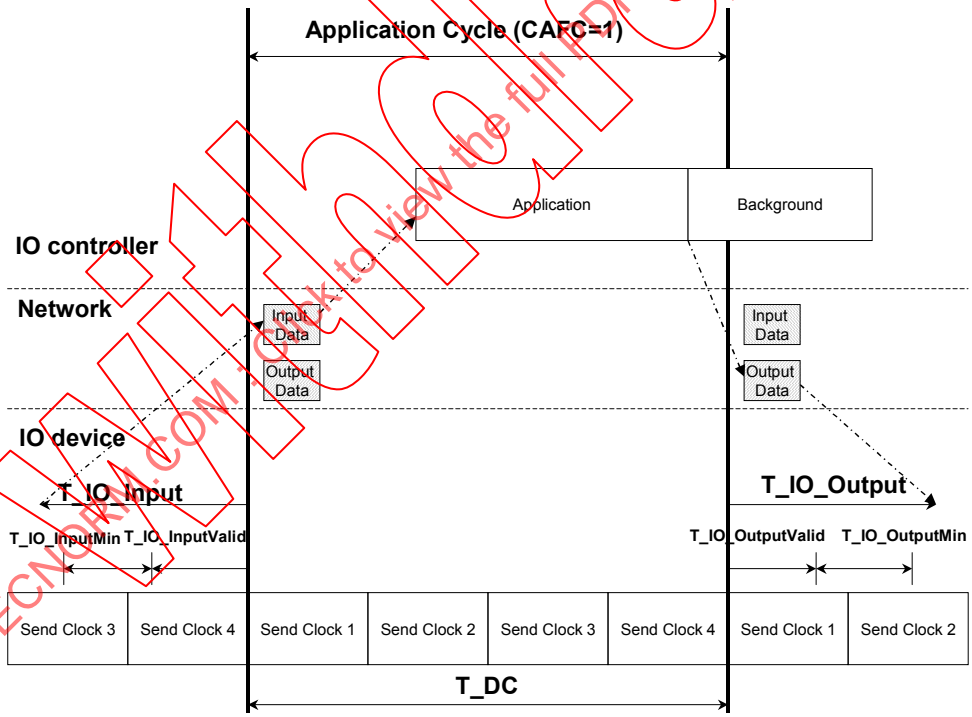


Figure 91 – General isochronous application model (example)

8.3.7.3 Isochronous Mode Application service specification

8.3.7.3.1 Write IsoM Data

This confirmed service may be used to write the attributes of the IsoM Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 314 shows the parameters of the service.

The service user shall only address modules and submodules within the List of Modules, which are part of the submodule list the AR is connected to. Otherwise the service provider shall issue a negative response to the service and all data shall be ignored.

The service parameter “Multiple” may be used to convey multiple IsoM Data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE 1 It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 314 – Write IsoM Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
IsoM Data	M	M(=)		
Time Data Cycle	M	M(=)		
Time IO Input	M	M(=)		
Time IO Output	M	M(=)		
Time IO Input Valid	M	M(=)		
Time IO Output Valid	M	M(=)		
Controller Application Cycle Factor	M	M(=)		
Prm Flag		M		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Slot Number
The parameter Slot Number is used to address a specific slot.

Subslot Number
The parameter Subslot Number is used to address a specific subslot.

Multiple
The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value

MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16} -1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to 2^{32}-256.

IsoM Data
This parameter is composed of the following elements:

Time Data Cycle
This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Input
This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Output
This parameter contains the value corresponding attribute of the ASE object.

Time IO Input Valid
This parameter contains the value corresponding attribute of the ASE object.

Time IO Output Valid
This parameter contains the value corresponding attribute of the ASE object.

Controller Application Cycle Factor
This parameter contains the value corresponding attribute of the ASE object.

Prm Flag
The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Type: Boolean

NOTE 2 This parameter is used to control the persistent storage of data. It is utilized in fast start-up procedure.

Result(+)
This parameter indicates that the service request succeeded.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 3 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.7.3.2 Read IsoM Data

This confirmed service may be used to read the values of the attributes of the IsoM Data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 315 shows the parameters of the service.

Implicit AR using Target AR UUID: This service does only contain Real Port Data if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 315 – Read IsoM Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
IsoM Data			M	M(=)
Time Data Cycle			M	M(=)
Time IO Input			M	M(=)
Time IO Output			M	M(=)
Time IO Input Valid			M	M(=)
Time IO Output Valid			M	M(=)
Controller Application Cycle Factor			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error code 1			M	M(=)
Error code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument
The argument shall convey the service specific parameters of the service request.

AREP
This parameter is the local identifier for the desired AR.

API
This parameter shall be used to address the desired API.

Target AR UUID
This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE 1 It is used to read only port difference information the requested AR is connected to.

Slot Number
The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number
The parameter Subslot Number is used to address port information for a specific subslot.

Seq Number
The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to 2^{16}-1. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length
The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to 2^{32}-256.

Result(+)
This parameter indicates that the service request succeeded.

IsoM Data
This parameter is composed of the following elements:

Time Data Cycle
This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Input
This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Output
This parameter contains the value corresponding attribute of the ASE object.

Time IO Input Valid
This parameter contains the value corresponding attribute of the ASE object.

Time IO Output Valid
This parameter contains the value corresponding attribute of the ASE object.

Controller Application Cycle Factor
This parameter contains the value corresponding attribute of the ASE object.

Result(-)
This parameter indicates that the service request failed.

Error Decode
This parameter selects one Error scheme for Error code 1 and 2; its coding is specified in IEC 61158–6–10.

Type: Unsigned8

Allowed Value: PNIORW

Error code 1
The Error code 1 assumes one of the following values:
read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error code 2
The parameter Error code 2 is user specific.

Type: Unsigned8

Add Data 1
The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE 2 Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2
The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE 3 Add Data 2 may be used by device vendors to transmit specific error messages.

8.3.7.3.3 SYNCH Event

The service SYNCH indicates to the application of the IO device that a new isochronous send clock cycle has been started.

Table 316 shows the parameters of the service.

Table 316 – SYNCH Event

Parameter name	Ind
Argument	M
Global Cycle Counter	M
Status	M

Argument
The argument shall convey the service specific parameters of the service indication.

Global Cycle Counter
This parameter shall contain the local value of the cycle counter.

Status
This parameter shall contain the current status. The allowed values are LOCAL and REMOTE. The value LOCAL shall be used to indicate that no valid synchronization frame has been received within the PLL window. The value REMOTE shall be used to indicate that a valid synchronization frame has been received within the PLL window.

8.3.7.4 Behavior of Isochronous Mode Application process objects

8.3.7.4.1 Overview of relations to other ASE objects

For performance reasons, the value for the Output Data object shall be transferred on receipt as fast as possible to the object. The speed and the jitters of this action is a performance parameter of an IO device. This parameter determines the synchronization accuracy.

The Isochronous Mode Application model is a superset of the model described in IEC 61158–5–3.

Figure 92 shows the timing relations between different involved ASEs.

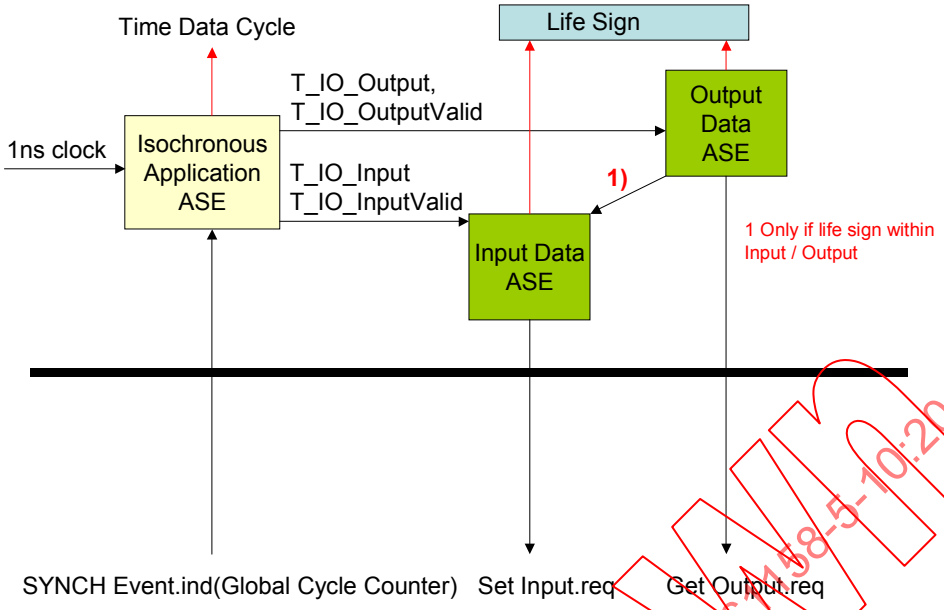


Figure 92 – ASE relations in an IO device operating in isochronous mode

Figure 93 shows the state machine relations between different involved ASEs.

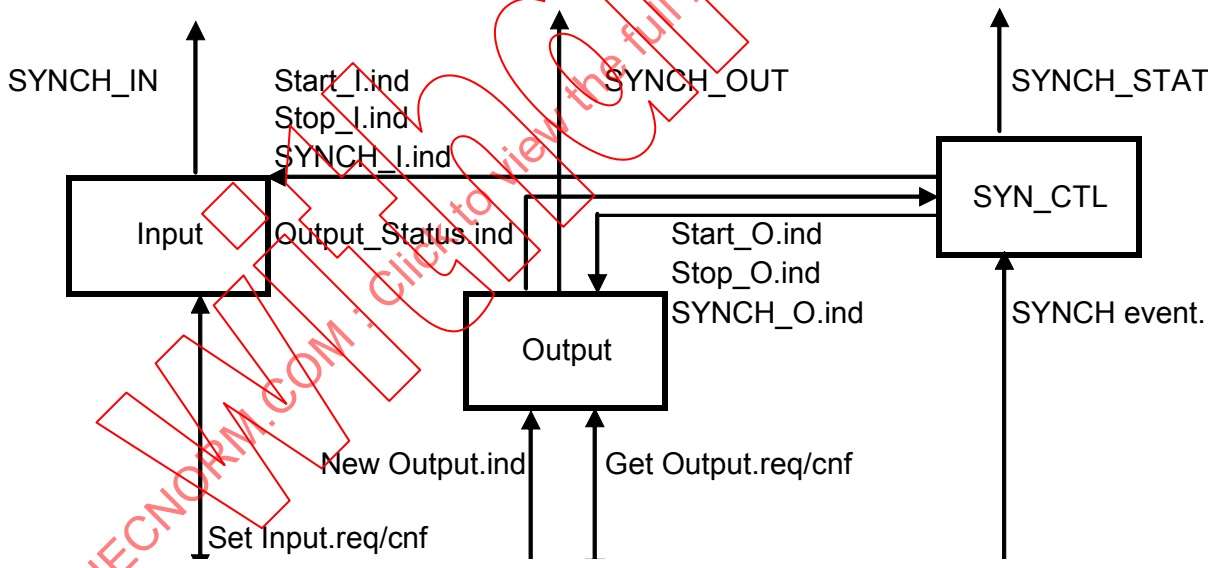


Figure 93 – State machine relations in an IO device operating in isochronous mode

8.3.7.4.2 Basic behavior of input and output data in the isochronous mode

An IO system operating in the isochronous mode provides the following features:

- all IO devices taking part in the isochronous mode functionality synchronize their internal clock system with the synchronization message SYNCH Event with the isochronous data cycle
- all IO devices taking part in the isochronous mode functionality have to be in the same PTCP subdomain and furthermore in the same isochronous real-time domain
- the jitter of SYNCH Event.ind is less than 1 µs (jitters within an IO device is assumed to be zero)
- in each data cycle at least one Ethernet frame of full length may be included

In an IO system with enabled isochronous mode functionality IO devices with basic functionality can be used in conjunction with synchronized IO devices:

- if the SYNCH-Event.ind is issued with the status “REMOTE” then the application is synchronized with other devices
- if the SYNCH-Event.ind is issued with the status “LOCAL” then the isochronous application shall be informed that the application may be not synchronized with other devices

8.3.7.4.3 State machine description for isochronous mode

8.3.7.4.3.1 General

The behavior of an Output Data object is defined by the Output Data and SyncCtl state machine in case of Isochronous Mode.

The Output Data and SyncCtl state machine represents a part of the User functionality and are included to ensure interoperability.

The Output Data and SyncCtl state machine provides events to other parts of the User functionality. The SyncCtl state machine has interfaces to the Input Data state machine and the Output Data state machine for co-ordination purposes.

The main functionalities of the Output Data state machine are:

- Check the correct sequence: SYNCH, New Output, read out the Output Data at Time IO Output.
- Transfer the received Output Data to the application on time.
- Check the Masters Sign of Life (MLS).

The main functionalities of the SyncCtl state machine are:

- Generate timing signals from the SYNCH-Event.ind
- Generation of a pulse scheme for the local application

The synchronization primitive (SYNCH Event) is conveyed by a local PLL.

8.3.7.4.3.2 Primitive definitions

8.3.7.4.3.2.1 Primitives exchanged between the AL and the SyncCtl state machine

Table 317 shows primitive the exchanged between the AL and the SyncCtl state machine.

Table 317 – Primitives issued by the AL to the SyncCtl state machine

Primitive Name	Source	Associated parameters	Functions
SYNCH Event.ind	AL	AREP Global Cycle Counter Status	
Started.ind	AL	AREP	
Stopped.ind	AL	AREP	

The parameters used with the primitives are described in the Service Specification of the Output Data object (see 4.3.3.2.2.3).

8.3.7.4.3.2.2 Primitive exchanged between the SyncCtl state machine and the user

Table 318 shows the primitive exchanged between the SyncCtl state machine and the user.

Table 318 – Primitive issued by the SyncCtl state machine to the user

Primitive Name	Source	Associated parameters	Functions
SYNCH_STATUS	SyncCtl	Status	Reports change of Operation Mode, Warnings and Errors

8.3.7.4.3.2.3 Primitive exchanged between the Input state machine and the user

Table 319 shows the primitive exchanged between the Input state machine and the user.

Table 319 – Primitives issued by the Input state machine to the user

Primitive Name	Source	Associated parameters	Functions
SYNCH_IN	Input		Collects the Values of the Input User Data

8.3.7.4.3.2.4 Primitive exchanged between the output state machine and the user

Table 320 shows the primitive exchanged between the Output state machine and the user.

Table 320 – Primitive issued by the Output state machine to the user

Primitive Name	Source	Associated parameters	Functions
SYNCH_OUT	Output		Pass the Output Buffer to the Output User Data

8.3.7.4.3.2.5 Primitives exchanged between the Output state machine and the SyncCtl state machine

Table 321 and Table 322 show the primitive exchanged between the Output state machine and the SyncCtl state machine.

Table 321 – Primitives issued by the SyncCtl to the output state machine

Primitive Name	Source	Associated parameters	Functions
Start_O.ind	SyncCtl		Start Output Data processing
Stop_O.ind	SyncCtl		Stop Output Data processing
SYNCH_O.ind	SyncCtl		Trigger Signal for Output Data Processing

Table 322 – Primitives issued by the output to the SyncCtl state machine

Primitive Name	Source	Associated parameters	Functions
Output_Status.ind	Output	Status	Report Status of Output Data processing

The parameter Status used with the primitive Output_Status.ind reports the different warnings and errors.

8.3.7.4.3.2.6 Primitives exchanged between the Input state machine and the SyncCtl state machine

Table 323 shows the primitive exchanged between the Input state machine and the SyncCtl state machine.

Table 323 – Primitives issued by the SyncCtl to the input state machine

Primitive Name	Source	Associated parameters	Functions
Start_I.ind	SyncCtl		Start Input Data processing
Stop_I.ind	SyncCtl		Stop Input Data processing
SYNCH_I.ind	SyncCtl	Status	Trigger Signal for Input Data Processing

8.3.7.4.3.2.7 Primitives exchanged between the AL and the Output state machine

Table 324 shows the primitive exchanged between the Output state machine and AL.

Table 324 – Primitives issued by the output state machine to the AL

Primitive Name	Source	Associated parameters	Functions
Get_Output.req	Output	AREP	

Table 325 shows the primitive exchanged between the AL and Output state machine.

Table 325 – Primitives issued by the AL to the output state machine

Primitive Name	Source	Associated parameters	Functions
Get_Output.cnf	AL	AREP IOPS Subslot_Output_Data New_Flag IOCS	Confirms Get_Output request and passes new output data to output state machine
New_Output.ind	AL	AREP CREP Slot Number Subslot Number WatchdogFlag InDataFlag	Indicates new output data has been received

8.3.7.4.3.2.8 Primitives exchanged between the Input state machine and the AL

Table 326 shows the primitive exchanged between the Input state machine and the AL.

Table 326 – Primitives issued by the input state machine to the AL

Primitive Name	Source	Associated parameters	Functions
Set_Input.req	Input	AREP CREP Slot Subslot IOPS Input_Data	Passes application input data to AL

Table 327 shows the primitive exchanged between the Input state machine and the AL.

Table 327 – Primitives issued by the AL to the input state machine

Primitive Name	Source	Associated parameters	Functions
Set_Input.cnf	AL	AREP	Confirms Set_Input request

8.3.7.4.3.3 SyncCtl state diagram

Figure 94 shows the state diagram of the SyncCtl state machine.

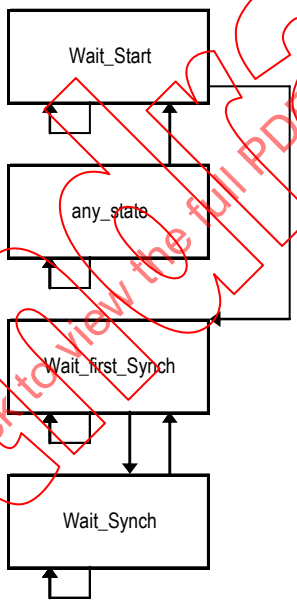


Figure 94 – SyncCtl state diagram

8.3.7.4.3.4 SyncCtl state table

Table 328 shows the states of a SyncCtl state machine.

Table 328 – SyncCtl state table

#	Current State	Event /Condition =>Action	Next State
1	Wait_Start	**SYNCH Event.ind(AREP, GlobalCycleCounter, Status)** => ignore	Wait_Start
2	any_state	**Started.ind(AREP)** => RUN := 0	Wait_first_SYNCH
3	any_state	**Stopped.ind(AREP)** => O_Status := OUTPUT_RESETMLS SYNCH Status.ind(Status)	Wait_Start_PLL
4	any_state	**Output Status.req(Status)** => O_Status := Status	SAME
5	Wait_first_Sync	**SYNCH Event.ind(AREP, GlobalCycleCounter, Status)** /Status == LOCAL =>	Wait_first_Sync
6	Wait_first_Sync	**SYNCH Event.ind(AREP, GlobalCycleCounter, Status)** /Status != LOCAL && (GlobalCycleCounter mod SendClock*ReductionRatio) != 0 =>	Wait_first_Sync
7	Wait_first_Sync	**SYNCH Event.ind(AREP, GlobalCycleCounter, Status)** /Status != LOCAL && (GlobalCycleCounter mod SendClock*ReductionRatio) == 0 => Start_O.ind SYNCH_I.ind SYNCH_O.ind SYNCH_Status.ind(Status)	Wait_Synch
8	Wait_Synch	**SYNCH Event.ind(AREP, GlobalCycleCounter, Status)** /Status == LOCAL => SYNCH_Status.ind(Status) Stop_I.ind	Wait_first_Sync
9	Wait_Synch	**SYNCH Event.ind(AREP, GlobalCycleCounter, Status)** /Status != LOCAL && (GlobalCycleCounter mod SendClock*ReductionRatio) != 0 =>	Wait_Synch
10	Wait_Synch	**SYNCH Event.ind(AREP, GlobalCycleCounter, Status)** /Status != LOCAL && (GlobalCycleCounter mod SendClock*ReductionRatio) == 0 => Start_O.ind SYNCH_I.ind SYNCH_O.ind SYNCH_Status.ind(Status)	Wait_Synch

8.3.7.4.3.5 Output state diagram

Figure 95 shows the state diagram of the Output state machine.

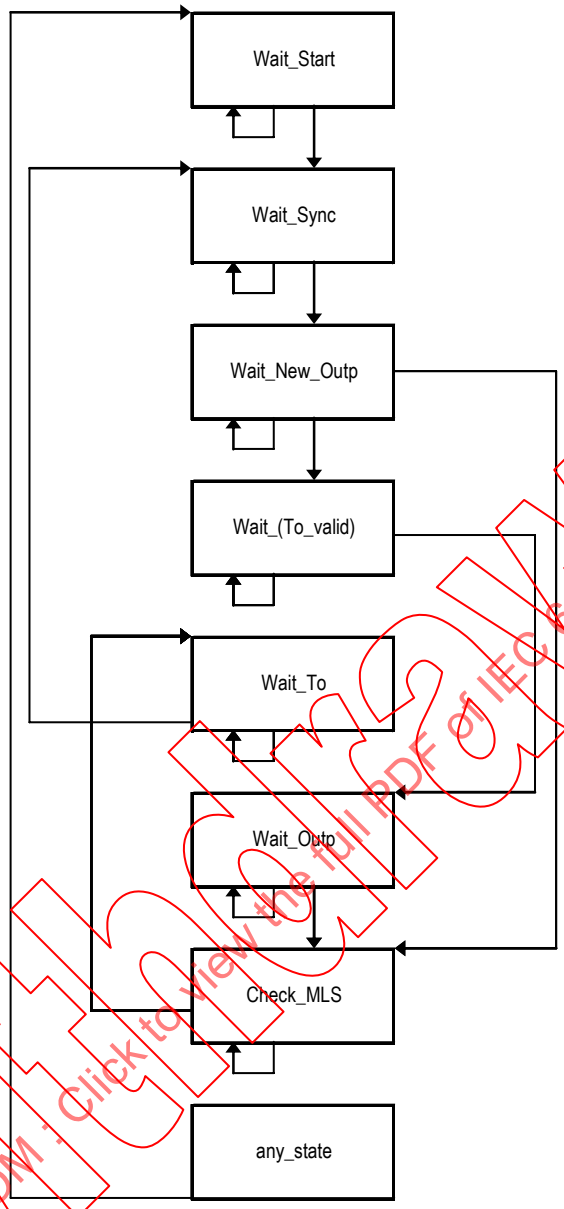


Figure 95 – Output state diagram

8.3.7.4.3.6 Output state table

Table 329 shows the states of an Output state machine.

Table 329 – Output state table

#	Current State	Event /Condition =>Action	Next State
1	Wait-Start	**SynchO.req** => Start Timer(To_valid) Start TimerTo	Wait_New_Outp
2	Wait-Start	**New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag)** /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Wait-Start
3	Wait-Start	**New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag)** /wrong_output_upd >= limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Wait-Start
4	Wait_New_Outp	**New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag)** => if (wrong_output_upd>0) wrong_output_upd= wrong_output_upd-1	Wait_(To_valid)
5	Wait_New_Outp	**Timer(To_valid) expired** /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Check_MLS
6	Wait_New_Outp	**Timer(To_valid) expired** /wrong_output_upd >= limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Check_MLS
7	Wait_(To_valid)	**Timer(To_valid) expired** => Get_Output.req(AREP)	Wait_Outp
8	Wait_(To_valid)	**New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag)** /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Wait_(To_valid)
9	Wait_(To_valid)	**New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag)** /wrong_output_upd >= limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Wait_(To_valid)
10	Wait_To	**TimerTo expired** => SYNCH_Out	Wait-Start
11	Wait_To	**New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag)** /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Wait_To

#	Current State	Event /Condition =>Action	Next State
12	Wait_To	**New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag)** /wrong_output_upd >= limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Wait_To
13	any_state	**Start_O.req** => MLS_State:= Wait_first_MLS O_Buffer:=Nil(with O_Buffer.MLS:=0) old_MLS:=0 MLS_error_counter:=0 TMAPC_counter:=0 MLS_start_counter:=0	Wait-Start
14	Wait_Outp	**Get_Output.cnf (AREP, IOPS, Subslot_Output_Data, New_Flag, IOCS)** /New_Flag && IOPS == GOOD => O_Buffer := Output_Data	Check_MLS
15	Wait_Outp	**Get_Output.cnf (AREP, IOPS, Subslot_Output_Data, New_Flag, IOCS)** /!(New_Flag && IOPS == GOOD) =>	Check_MLS
16	Check_MLS	/MLS_State = Wait_first_MLS => MLS_State := Wait_next_MLS Status:=OUTPUT_RESETMLS old_MLS:=O_Buffer.MLS Output_Status.ind (Status)	Wait_To
17	Check_MLS	/MLS_State = Wait_next_MLS TMAPC_counter < MLS_factor => MLS_State := Wait_next_MLS inc(TMAPC_counter)	Wait_To
18	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter < n_TMAPC => MLS_State := Wait_next_MLS inc_LS(old_MLS) TMAPC_counter:=0 MLS_error_counter := MLS_error_counter + n_error	Wait_To

#	Current State	Event /Condition =>Action	Next State
19	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter >= n_TMAPC => MLS_State := Wait_first_MLS Status:=OUTPUT_RESETMLS TMAPC_counter:=0 MLS_error_counter:=0 old_MLS:=0 MLS_start_counter:=0 Output_Status.ind (Status)	Wait_To
20	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS = inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_start_counter < n_MLS => MLS_State := Wait_next_MLS TMAPC_counter:=0 if (MLS_error_counter >0) MLS_error_counter := MLS_error_counter -1 old_MLS:=MLS inc(MLS_start_counter)	Wait_To
21	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS = inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_start_counter = n_MLS => MLS_State := Wait_MLS Status:=OUTPUT_RUNMLS TMAPC_counter:=0 MLS_error_counter:=0 old_MLS:=O_Buffer.MLS MLS_start_counter:=0 Output_Status.ind (Status)	Wait_To
22	Check_MLS	/MLS_State = Wait_MLS TMAPC_counter < MLS_factor => MLS_State := Wait_MLS inc(TMAPC_counter)	Wait_To
23	Check_MLS	/MLS_State = Wait_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter < n_TMAPC => MLS_State := Wait_MLS inc_LS(old_MLS) TMAPC_counter:=0 MLS_error_counter := MLS_error_counter + n_error	Wait_To

#	Current State	Event /Condition =>Action	Next State
24	Check_MLS	/MLS_State = Wait_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter >= n_TMAPC => MLS_State := Wait_first_MLS Status:=OUTPUT_RESETMLS TMAPC_counter:=0 MLS_error_counter:=0 old_MLS:=0 Output_Status.ind (Status)	Wait_To
25	Check_MLS	/MLS_State = Wait_MLS O_Buffer.MLS = inc_LS(old_MLS) & TMAPC_counter = MLS_factor => MLS_State := Wait_MLS TMAPC_counter:=0 if (MLS_error_counter >0) MLS_error_counter := MLS_error_counter -1 old_MLS:=O_Buffer.MLS	Wait_To

8.3.7.4.3.7 Input state diagram

Figure 96 shows the state diagram of the Input state machine.

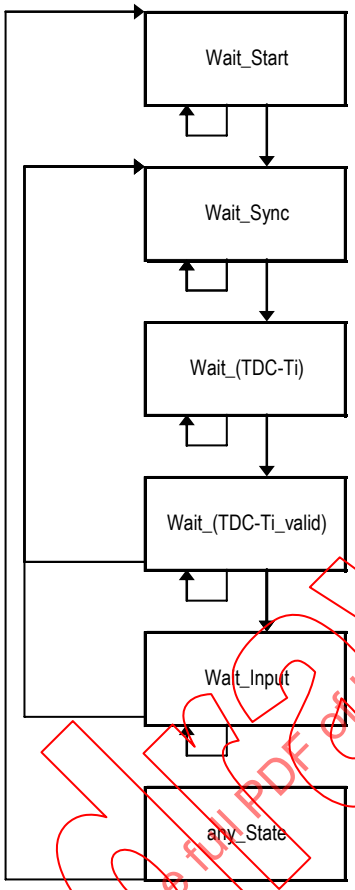


Figure 96 – Input state diagram

8.3.7.4.3.8 Input state table

Table 330 shows the states of an Input state machine.

Table 330 – Input state table

#	Current State	Event /Condition =>Action	Next State
1	Wait-Start	**Start_I.ind** =>	Wait-Sync
2	Wait-Sync	**SYNCH_I.ind(Output_State)** => Start Timer(TDC-Ti) Start TimerTix Store Output_State	Wait_(TDC-Ti)
3	Wait_(TDC-Ti)	**Timer(TDC-Ti) expired** => SYNCH_In	Wait_Tix
4	Wait_Tix	**TimerTix expired** /LS in Submodule && Output_State == OUTPUTRUNMLS => Input_Data:= I_Buffer inc_LS(SLS) Input_Data.SLS:= SLS Set_Input.req (AREP,CREP, Slot, Subslot, IOPS, Input_Data)	Wait_Input
5	Wait_Tix	**TimerTix expired** /LS in Submodule && !(Output_State == OUTPUTRUNMLS) => Input_Data:= I_Buffer SLS := 0 Input_Data.SLS:= SLS Set_Input.req (AREP,CREP, Slot, Subslot, IOPS, Input_Data)	Wait_Input
6	Wait_Tix	**TimerTix expired** /!LS in Submodule => Input_Data:= I_Buffer Set_Input.req (AREP,CREP, Slot, Subslot, IOPS, Input_Data)	Wait_Input
7	Wait_Input	**Set_Input.cnf(AREP)** =>	Wait-Sync
8	any_State	**Stop_I.ind** =>	Wait-Start

8.3.8 Physical Device Management ASE

8.3.8.1 Overview

This ASE specifies the structure of the necessary parameters for the used protocols, e.g. ISO/IEC 8802-3, IP, UDP, RPC, or DCP, which are inherit by common ASEs. The Physical Device Management ASE manages the parameter in a common database independent of the device instances. It represents the physical device as shown in Figure A.1. The Physical Device Management ASE provides a set of services used to read, and write their values.

The attribute Station Name within the DCP group is essential for the function of an IO device and an IO controller. It shall be set by means of the DCP service DCP Set or local means. The IO device and IO controller shall store this attribute persistent. Without a Station Name (Chassis ID) the operation of an IO device is not permitted.

The attributes may be volatile or persistent. Volatile attributes are invalid after a power cycle. The values of persistent attributes remain after a power cycle.

There are two variants defined obtaining an IP address. Common to all variants is using the Station Name (Chassis ID) as the key for selecting the IP address.

The DCP protocol is mandatory and provides a service to set an IP address according to the Station Name (Chassis ID). An IO device obtains the IP address from the IO controller or IO supervisor via the DCP Set service. An IO device shall support a volatile IP address in general, and additionally a non-volatile IP address for fast start-up mode. The DCP server is only passive in this procedure.

NOTE In order to support legacy implementations an additional mode with permanent IP addresses is permissible.

An IO controller obtains the IP address from the IO supervisor via the DCP Set service, local means, or DHCP. An IO controller shall support a persistent IP address. But it shall store the IP address only persistent if it has not received the IP address over DHCP.

The DHCP protocol should also be supported. It provides

- unique IP addresses for large systems and
- IP access for other applications (e.g. WEB access) to the IO device in case the IO controller is switched off

The DHCP client requests the IP address using the selected options. The used DHCP options are provided by the DCP Set service or local means.

EXAMPLE 1 A small environment consists of one IO controller and two IO devices. Installing and configuring of a DHCP server is not appropriate in this case. Each device has got a unique station name. The IO controller identifies each IO device by means of the DCP Identify service with the filter station name and assigns its IP address with the DCP Set service.

EXAMPLE 2 A large environment consists of 27 IO controller and 64 IO devices per IO controller. The devices may be connected to different subnets. Installing and configuring of a DHCP server is appropriate in this case. Each device has got a unique station name and the use of DHCP is enabled. Each device requests its IP address from the DHCP server using the option e.g. Client Identifier = station name. The IO controller resolves the IP address of its related IO devices via DCP Identify or via DNS Get Host By Name service. The use of DCP Identify is restricted to a subnet.

The IO controller shall support DCP. Furthermore, it shall support DNS and DHCP. DNS is used to provide IP addresses to:

- access IO devices behind IP routers
- use IO devices which receive their IP addresses from a DHCP server
- get an IP address for IO devices (without DHCP) if the IP address is not part of engineering information and DCP is selected to provide the IP address to the IO device

The startup procedure for an IO device if the station name exists provides the following steps:

- start MAC interface
- start LLDP
- start DCP receiver and DHCP client (if DHCP enabled)
- wait for IP address

The startup procedure for an IO controller if the station name exists provides the following steps:

- start MAC interface
- start LLDP
- start DCP receiver, DCP sender, and DHCP client (if DHCP enabled)
- wait for IP address or use local configured IP address
- check its own IP address and station name for multiple use by means of ARP and DCP before usage (error duplicate IP address or duplicate station name)

The fast startup procedure for an IO device where the station name, IP address already exists provides the following steps:

- start MAC interface (auto-negotiation off)
- start LLDP
- start DCP receiver and sender, send DCP hello if configured
- wait for connection establishment

The IO controller provides the following steps to fast startup a related IO device:

i) wait for DCP hello indication, if configured
ii) CM Connect using stored IP address

The detailed procedure for startup is provided in the protocol specification.

Furthermore, the Physical Device Management ASE defines the object attributes and access methods specific for common parameter of interface and port submodules. Besides the address resolution and neighborhood detection, these submodules contain additional attributes for global clock synchronization and RT_CLASS_3 routing information. All this attributes shall be stored non-volatile for fast startup procedure.

8.3.8.2 Physical Device Management class specification

8.3.8.2.1 General

The Physical Device Management ASE defines one Physical Device Management object type:

8.3.8.2.2 Physical Device Management class specification

8.3.8.2.2.1 Template

A Physical Device Management object is described by the following template:

ASE: **Physical Device Management ASE**
CLASS: **Physical Device Management**
CLASS ID: **not used**
PARENT CLASS: **IEEE 802.3 IEEE 802.1AB IEEE 802.1D IP suite DNS DHCP SNMP DCP Media redundancy PTCP**
ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	Real List of Interfaces
2.1	(m)	Attribute:	Slot Number
2.2	(m)	Attribute:	Subslot Number
2.3	(m)	Attribute:	Real List of Ports
2.3.1	(m)	Attribute:	Slot Number
2.3.2	(m)	Attribute:	Subslot Number
2.3.3	(o)	Attribute:	MRP Domain UUID
2.3.4	(o)	Attribute:	MRP Domain UUID Adjusted

2.3.5	(m)	Attribute:	Domain Boundary
2.3.5.1	(m)	Attribute:	Ingress
2.3.5.2	(m)	Attribute:	Egress
2.3.6	(m)	Attribute:	Multicast Boundary
2.3.7	(m)	Attribute:	Peer To Peer Boundary
2.3.8	(m)	Attribute:	DCP Boundary
2.4	(m)	Attribute:	Expected List of Ports
2.4.1	(m)	Attribute:	Slot Number
2.4.2	(m)	Attribute:	Subslot Number
2.5	(m)	Attribute:	IR Data
2.5.1	(m)	Attribute:	List of RT_CLASS_3 Interfaces
2.5.1.1	(m)	Attribute:	Slot Number
2.5.1.2	(m)	Attribute:	Subslot Number
2.6	(m)	Attribute:	Real Sync Data
2.6.1	(m)	Attribute:	List of Interfaces
2.6.1.1	(m)	Attribute:	Slot Number
2.6.1.2	(m)	Attribute:	Subslot Number
2.6.1.3	(m)	Attribute:	PTCP Subdomain ID
2.6.1.4	(m)	Attribute:	PTCP Subdomain Name
2.6.1.5	(m)	Attribute:	IR Data ID
2.6.1.6	(m)	Attribute:	Reserved Interval Begin
2.6.1.7	(m)	Attribute:	Reserved Interval End
2.6.1.8	(m)	Attribute:	PLL Window
2.6.1.9	(m)	Attribute:	Sync Send Factor
2.6.1.10	(m)	Attribute:	Send Clock Factor
2.6.1.11	(m)	Attribute:	Sync Properties
2.6.1.11.1	(m)	Attribute:	Role
2.6.1.11.2	(m)	Attribute:	Sync Class
2.6.1.12	(m)	Attribute:	Sync Frame Address
2.6.1.13	(m)	Attribute:	PTCP Timeout Factor
2.7	(m)	Attribute:	Expected Sync Data
2.7.1	(m)	Attribute:	List of Interfaces
2.7.1.1	(m)	Attribute:	Slot Number
2.7.1.2	(m)	Attribute:	Subslot Number
2.7.1.3	(m)	Attribute:	PTCP Subdomain ID
2.7.1.4	(m)	Attribute:	PTCP Subdomain Name
2.7.1.5	(m)	Attribute:	IR Data ID
2.7.1.6	(m)	Attribute:	Reserved Interval Begin
2.7.1.7	(m)	Attribute:	Reserved Interval End
2.7.1.8	(m)	Attribute:	PLL Window
2.7.1.9	(m)	Attribute:	Sync Send Factor
2.7.1.10	(m)	Attribute:	Send Clock Factor
2.7.1.11	(m)	Attribute:	Sync Properties
2.7.1.11.1	(m)	Attribute:	Role
2.7.1.11.2	(m)	Attribute:	Sync Class
2.7.1.12	(m)	Attribute:	Sync Frame Address
2.7.1.13	(m)	Attribute:	PTCP Timeout Factor
2.8	(o)	Attribute:	Real Fiber Optic Data
2.8.1	(m)	Attribute:	List of Ports
2.8.1.1	(m)	Attribute:	Slot Number
2.8.1.2	(m)	Attribute:	Subslot Number
2.8.1.3	(m)	Attribute:	Fiber Optic type
2.8.1.4	(m)	Attribute:	Fiber Optic Cable type

2.8.1.5	(o)	Attribute:	Fiber Optic Manufacturer Specific
2.8.1.5.1	(m)	Attribute:	Vendor ID
2.8.1.5.3	(m)	Attribute:	Manufacturer Specific Fiber Optic Data
2.8.1.6	(m)	Attribute:	Maintenance Demanded Power Budget
2.8.1.7	(m)	Attribute:	Maintenance Required Power Budget
2.8.1.8	(m)	Attribute:	Error Power Budget
2.9	(o)	Attribute:	MRP Interface Data
2.9.1	(m)	Attribute:	List of MRP Domains
2.9.1.1	(m)	Attribute:	Real Data
2.9.1.2	(o)	Attribute:	Adjust Data
2.9.1.3	(o)	Attribute:	Check Enable
2.9.1.3.1	(m)	Attribute:	Multiple Manager Check
2.9.1.3.2	(m)	Attribute:	Domain UUID Check
2.10	(o)	Attribute:	List of Expected Fast Startup Data
2.10.1	(o)	Attribute:	Hello Data
2.10.1.1	(m)	Attribute:	Hello Mode
2.10.1.2	(m)	Attribute:	Hello Interval
2.10.1.3	(m)	Attribute:	Hello Retry
2.10.1.4	(m)	Attribute:	Hello Delay
2.10.2	(o)	Attribute:	Parameter Data
2.10.2.1	(m)	Attribute:	Parameter Mode
2.10.2.2	(m)	Attribute:	Parameter UUID
2.10.3	(o)	Attribute:	ARUUID
2.11	(o)	Attribute:	Network Component Data
2.11.1	(m)	Attribute:	List of Ports
2.11.1.1	(m)	Attribute:	Slot Number
2.11.1.2	(m)	Attribute:	Subslot Number
2.11.1.3	(m)	Attribute:	Maintenance Demanded Drop Budget
2.11.1.4	(m)	Attribute:	Maintenance Required Drop Budget
2.11.1.5	(m)	Attribute:	Error Drop Budget
2.12	(m)	Attribute:	Real Interface Data
2.12.1	(m)	Attribute:	Chassis ID
2.12.2	(m)	Attribute:	MAC Address
2.12.3	(m)	Attribute:	IP Address
3	(m)	Attribute:	Write Persitence Flag for Real List of Interfaces

SERVICES:

1	(m)	OpsService:	Write Expected Port Data
2	(m)	OpsService:	Write Adjusted Port Data
3	(m)	OpsService:	Read Real Port Data
4	(m)	OpsService:	Read Expected Port Data
5	(m)	OpsService:	Read Adjusted Port Data
6	(m)	OpsService:	Write IR Data
7	(m)	OpsService:	Read IR Data
8	(m)	OpsService:	Write Sync Data
9	(m)	OpsService:	Read Real Sync Data
10	(m)	OpsService:	Read Expected Sync Data
11	(m)	OpsService:	Read PDev Data
12	(m)	OpsService:	Sync Event
13	(m)	OpsService:	Write Fiber Optic Data
14	(m)	OpsService:	Read Real Fiber Optic Data
15	(m)	OpsService:	Write Interface MRP Data
16	(m)	OpsService:	Read Interface MRP Data
17	(m)	OpsService:	Write Port MRP Data

18 (m) OpsService: Read Port MRP Data
19 (m) OpsService: Write FSU Data
20 (m) OpsService: Read FSU Data
21 (m) OpsService: Write Network Component Data
22 (m) OpsService: Read Network Component Data
23 (m) OpsService: Read Real Interface Data

8.3.8.2.2.2 Attributes

Implicit
The attribute Implicit indicates that the PDMgmt object is implicitly addressed by the service.

Real List of Interfaces
This attribute list contains the following attributes and inherits attributes from parent class IEEE 802.1AB:

Slot Number
This attribute contains the slot number of the module containing the interface submodule.

Attribute type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number
This attribute contains the subslot number of the interface.

Attribute type: Unsigned16

Allowed Values shall be used according to Table 331.

Table 331 – Subslot number for interface submodules

Value (hexadecimal)	Meaning
0x8000	First Interface Submodule
0x8100	Interface Submodule 2
0x8200	Interface Submodule 3
0x8300	Interface Submodule 4
0x8400	Interface Submodule 5
0x8500	Interface Submodule 6
0x8600	Interface Submodule 7
0x8700	Interface Submodule 8
0x8900	Interface Submodule 9
0x8A00	Interface Submodule 10
0x8B00	Interface Submodule 11
0x8C00	Interface Submodule 12
0x8D00	Interface Submodule 13
0x8E00	Interface Submodule 14
0x8F00	Interface Submodule 15

Real List of Ports
This attribute list contains the following attributes and inherits attributes from parent class IEEE 802.1AB:

NOTE 1 The subordinated attributes of Real List of Ports are accessible via SNMP and LLDP MIB.

Slot Number
This attribute contains the slot number of the module containing the port submodule.

Attribute type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number
This attribute contains the subslot number of the local port.

Attribute type: Unsigned16

Allowed Values shall be used according to Table 332. The value for the parameter “i” in Table 332 shall correspond with the related interface submodule. Each interface may contain up to 255 ports.

NOTE 2 The related interface module may be placed in a different slot.

Table 332 – Subslot number for port submodules

Value (hexadecimal)	Meaning
0x8i01	First Port Submodule (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i02	Port Submodule 2 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i03	Port Submodule 3 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i04	Port Submodule 4 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i05	Port Submodule 5 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i06	Port Submodule 6 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i07	Port Submodule 7 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i08	Port Submodule 8 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i09	Port Submodule 9 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i0A	Port Submodule 10 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i0B	Port Submodule 11 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i0C	Port Submodule 12 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i0D	Port Submodule 13 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i0E	Port Submodule 14 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i0F	Port Submodule 15 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i10	Port Submodule 16 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
…	…
0x8iFF	Port Submodule 255 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)

MRP Domain UUID

This optional attribute contains the MRP Domain UUID of the port if MRP is supported and enabled. It shall have the same value as MRP Domain UUID Adjusted after receiving a Write MRP Port Data indication otherwise the default value FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF shall be used. However, only two MRP ports of a switch are allowed to use this default value at the same time. A Read MRP Port Data addressing the MRP Domain UUID of other unconfigured MRP ports shall result in a negative response with error object non existent.

Attribute type: UUID

MRP Domain UUID Adjusted

This optional attribute contains the MRP Domain UUID adjusted from a client of the port if MRP is supported and enabled. While receiving a Write MRP Port Data indication the current value of the attribute MRP Domain UUID shall also be overwritten immediately. A Read MRP Port Data addressing MRP Domain UUID Adjusted of an unconfigured MRP port shall result in a negative response with error object non existent.

Attribute type: UUID

Domain Boundary
This attribute consists of the following attributes:

Ingress
This attribute contains the value of the ingress synchronization domain boundary. The value shall be a list of allowed values for each multicast group.

Allowed values: BLOCK_MULTICAST_ADDRESS_GROUP_0,
PASS_MULTICAST_ADDRESS_GROUP_0 ...
BLOCK_MULTICAST_ADRESS_GROUP_31
PASS_MULICAST_ADDRESS_GROUP_31

The multicast groups are defined for the following IEEE 802 addresses:

ADRESS_GROUP_0: 01-0E-CF-00-04-00, 01-0E-CF-00-04-20, 01-0E-CF-00-04-40, 01-0E-CF-00-04-80
ADRESS_GROUP_1: 01-0E-CF-00-04-01, 01-0E-CF-00-04-21, 01-0E-CF-00-04-41, 01-0E-CF-00-04-81

...

ADRESS_GROUP_31: 01-0E-CF-00-04-1F, 01-0E-CF-00-04-3F, 01-0E-CF-00-04-5F, 01-0E-CF-00-04-9F

Egress
This attribute contains the value of the egress synchronization domain boundary. The allowed values shall be the same as for the Ingress attribute.

Multicast Boundary
This attribute contains the indication of the multicast boundary.

Attribute type: Unsigned32

Peer to Peer Boundary
This attribute contains the indication of the LLDP and PTCP domain boundary.

Attribute type: Unsigned32

Allowed Values: PASS, BLOCK_LLDP, BLOCK_PTCP, BLOCK_BOOTH

DCP Boundary
This attribute contains the indication of the DCP multicast boundary.

Attribute type: Unsigned32

Allowed Values: PASS, BLOCK_HELLO, BLOCK_IDENTIFY, BLOCK_BOOTH

Expected List of Ports
This attribute list contains the same attributes as specified for Real List of Ports including all subsequent attributes.

IR Data
This attribute contains the following attributes:

List of RT_CLASS_3 Interfaces
This attribute contains the RT_CLASS_3 interfaces for IR Data Elements. A list element includes the following attributes and inherits attributes from parent class IEEE 802.1D:

Slot Number
This attribute contains the slot number of the module containing the interface submodule.

Attribute type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number
This attribute contains the subslot number of the interface.

Attribute type: Unsigned16

Allowed Values shall be used according to Table 333.

Table 333 – Subslot Number for Interface Submodules

Value (hexadecimal)	Meaning
0x8000	First Interface Submodule
0x8100	Interface Submodule 2
0x8200	Interface Submodule 3
0x8300	Interface Submodule 4
0x8400	Interface Submodule 5
0x8500	Interface Submodule 6
0x8600	Interface Submodule 7
0x8700	Interface Submodule 8
0x8900	Interface Submodule 9
0x8A00	Interface Submodule 10
0x8B00	Interface Submodule 11
0x8C00	Interface Submodule 12
0x8D00	Interface Submodule 13
0x8E00	Interface Submodule 14
0x8F00	Interface Submodule 15

Real Sync Data
This attribute contains the following attributes:

List of Interfaces
This attribute contains the Data Elements. A list element includes the following attributes:

Slot Number
This attribute contains the slot number of the module containing the interface submodule.

Attribute type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number
This attribute contains the subslot number of the interface.

Attribute type: Unsigned16

Allowed Values shall be used according to Table 334.

Table 334 – Subslot Number for Sync Interface Submodules

Value (hexadecimal)	Meaning
0x8000	First Interface Submodule
0x8100	Interface Submodule 2
0x8200	Interface Submodule 3
0x8300	Interface Submodule 4
0x8400	Interface Submodule 5
0x8500	Interface Submodule 6
0x8600	Interface Submodule 7
0x8700	Interface Submodule 8
0x8900	Interface Submodule 9
0x8A00	Interface Submodule 10
0x8B00	Interface Submodule 11
0x8C00	Interface Submodule 12
0x8D00	Interface Submodule 13
0x8E00	Interface Submodule 14
0x8F00	Interface Submodule 15

PTCP Subdomain ID
This attribute contains the unique identifier of the PTCP subdomain which is configured by project planning. A PTCP subdomain is a certain amount of IO controller and/or IO devices, which synchonize their send clocks.

Attribute type: UUID

PTCP Subdomain Name
This attribute contains the name of the PTCP subdomain which is configured by project planning. A PTCP subdomain is a certain amount of IO controller and/or IO devices, which synchonize their send clocks.

Attribute type: see Chassis ID

IR Data ID
This attribute contains the unique identifier of the isochronous realtime project which is configured by project planning.

Attribute type: UUID

Reserved Interval Begin
This attribute contains the start of RT_CLASS_2 reserved bandwith relatively to the current phase in nanoseconds.

Attribute type: Unsigned32

Reserved Interval End
This attribute contains the end of RT_CLASS_2 reserved bandwith relatively to the current phase in nanoseconds.

Attribute type: Unsigned32

PLL Window
This attribute contains the maximum jitter of the start of the isochrone cycle with a time base of 1 ns. The value 0 indicates no PLL Window defined within the Read Real Sync Data service.It is don't care for the Sync Master.

Attribute type: Unsigned32

Sync Send Factor
This attribute contains the send interval of the sync PDUs. The Sync Send Factor shall be set in multiples of 31,25 µs.

Attribute type: Unsigned16

The values shall be in the range of 1 to 2764800 000.

Send Clock Factor
This attribute is defined in 8.3.10.4.2.

Sync Properties
This attribute contains the following attributes:

Attribute type: Unsigned16

Role
This attribute contains the role of the device related to time synchronization.

Allowed values shall be according toTable 335.

Table 335 – Sync Properties Role

Meaning
Local sync
External sync: Role Sync Slave
Sync Master

Sync Class
This attribute contains the value for the Sync Class. Allowed values shall be according to Table 336.